Integrating precipitation nowcasting in a Deep Learning-based flash flood prediction framework and assessing the impact of rainfall forecasts uncertainties

Rim Mhedhbi

A THESIS SUBMITTED TO THE FACULTY OF GRADUATE STUDIES IN PARTIAL FULFILLMENT OF THE REQUIREMENTS FOR THE DEGREE OF MASTER OF ARTS

GRADUATE PROGRAM IN INFORMATION SYSTEMS AND TECHNOLOGIES

YORK UNIVERSITY
TORONTO, ONTARIO

May 2022

Abstract

Flash floods are among the most immediate and destructive natural hazards. To issue warnings on time, various attempts were made to extend the forecast horizon of flash floods prediction models. Particularly, introducing rainfall forecast into process-based hydrological models was found effective. However, integrating precipitation predictions into flash flood data-driven models has not been addressed yet. In this endeavor, we propose a modeling framework that integrates rainfall nowcasts and assesses the impact of rainfall predictions uncertainties on a Deep Learning-based flash flood prediction model. Compared to the Persistence and ARIMA models, the LSTM model provided better rainfall nowcasting performance. Further, we proposed an Encoder-Decoder LSTM-based model architecture for short-term flash flood prediction that supports rainfall forecasts. Computational experiments showed that future rainfall values improved flash floods' predictability for extended lead times. We also found that rainfall underestimation had a significant adverse effect on the model's performance compared to rainfall overestimation.

Keywords: flash flood, LSTM, uncertainty quantification, precipitation nowcasting

Acknowledgement

Firstly, I would like to express my sincere gratitude to Professor Marina G. Erechtchouk-ova for giving me the opportunity to accomplish this endeavor under her supervision. Professor Marina motivated me to give my best, continuously appraised my work, and provided me with valuable insights and feedback to improve my research and writing skills. Under her guidance, I was able to acquire valuable knowledge about hydrology, machine learning, and deep learning domains.

I extend my gratitude to the committee members Professor Xiaohui Yu and Professor Rashid Bashir, for taking the time to review and evaluate my work.

Finally, I dedicate this modest work to my Father, Bakey Mhedhbi, and my mother, Jamila Jarrar, who gave their best to bring me up and to whom I will indefinitely be in dept to. I thank my sisters Ahlam, Meriam, and Sahar for their continuous love and guidance. Special thanks to Ahmad who supported me during my thesis journey and helped me grow both personally and academically. I would love to extend my thanks to my friends who made my journey more enjoyable.

Table of Contents

List of Tables

List of Figures

Acronyms

ACF Auto Correlation Function. 28, 66, 67

AI Artificial Intelligence. 5

AMI Average Mutual Information. 40

ANFIS Adaptive Neuro-Fuzzy Inference System. 24, 29, 45

ANN Artificial Neural Network. 7, 8, 15, 27, 39–47

ARIMA Autoregressive Integrated Moving Average. 39, 41, 42, 44, 46, 47

ARMA Autoregressive Moving Average. 42

ARNN Autoregressive Neural Network. 40

BEMCA Bayesian Enhanced Modified Combined Approach. 40

BNN Bayesian Neural Networks. 13, 15

BPNN Back Propagation Neural Network. 30, 31, 39, 44, 45

CBP Continuous Basis Pursuit. 45

CCF Cross-Correlation Function. 28

CNN Convolutional Neural Network. 11, 47, 48

CTBT Cloud-Top Brightness Temperature. 48

DL Deep Learning. 10–12, 17, 46, 49, 50, 57, 58

DRCF Dynamic Regional Combined short-term rainfall Forecasting. 45

DT Decision Tree. 41, 42, 47

FC-LSTM Fully Connected-LSTM. 49

FFNN Feed Forward Neural Network. 8, 23, 29, 31, 39, 44, 45

FSP Floodwater Storage Pond. 30

GA Genetic Algorithm. 27, 29, 46

GNSS Global Navigation Satellite System. 43

GRU Gated Recurrent Unit. 25, 31, 48, 49

KDD Knowledge Discovery in Databases. v, 54

KNN k-Nearest Neighbors. 26, 40–42

LCA Linear Correlation Analysis. 40

LDA Linear Discriminant Analysis. 26

LogR Logistic Regression. 7, 41, 42

LR Linear Regression. 7, 31, 32, 40, 43, 48

LSTM Long Short-Term Memory. 23–25, 31, 32, 46–49

MA Moving Averages. 34, 40

MAE Mean Absolute Error. 64

MANN Modular Artificial Neural Network. 40

MAPE Mean Absolute Percentage Error. 64

MARIMA Multivariate Autoregressive Integrated Moving Average. 39

MC Monte Carlo. 15, 17

ML Machine Learning. 3, 5–7, 26, 38, 39, 41, 42, 46, 50, 55, 56

MLP Multilayer perceptron. 27, 29, 30, 32, 33, 41, 42, 45, 48

MLR Multiple Linear Regression. 29, 34

MRE Maximum Residual Error. 64

MSE Mean Squared Error. 45

MT Model Tree. 41

NARX Nonlinear Autoregressive Network with Exogenous inputs. 30

NWP Numerical Weather Prediction. 37–39, 42, 44, 46–49

PACF Partial Auto Correlation Function. 29, 66, 67

PCA Principal Component Analysis. 40, 46

PMI Partial Mutual Information. 40

PWV Precipitable Water Vapor. 43

RBFNN Radial Basis Function Neural Network. 39, 41, 46

RF Random Forest. 43, 44, 47

RMSE Root Mean Square Error. 23, 41, 45, 46, 64

RNN Recurrent Neural Network. 8, 11, 30, 31, 44–46, 48, 62

SARIMA Seasonal Autoregressive Integrated Moving Average. 40

SMAPE Symmetric Mean Absolute Percentage Error. 64

SSA Singular Spectrum Analysis. 40

SVM Support Vector Machine. 25, 26, 29, 32, 34, 40, 42, 43, 46

SVR Support Vector Regression. 22, 28, 29, 32, 47

TDNN Time Delay Neural Network. 39, 44

TLRNN Time Lagged Recurrent Neural Network. 41

TRCA Toronto and Region Conservation Authority. 58

TREC Tracking a Radar Echo by Cross-Correlation. 44

VAR Vector Autoregression. 39, 41, 44

VI Variational inference. 14, 15

XGBoost Extreme Gradient Boosting. 25, 34

Chapter 1

Introduction

Floods are among the most devastating natural hazards in the world. The immense moving power of their waters can cause mass destruction and debris deposition. These residues can contain toxic and contaminated chemicals, which can cause more lethal repercussions. Floods vary significantly depending on their causes, the size of affected areas, and the time required for flood development. Timely prediction of a flood event is a critical problem attracting the efforts of many researchers and practitioners. In particular, flash floods pose an additional challenge as these phenomena are characterized by their swift behavior. Flash floods are usually triggered by short intensive precipitations leaving minimal time for mitigation measures such as warnings, dam closures, and evacuation procedures. In fact, flash floods are considered the number-one weather-related killer in the US [12]. For example, in 2001, a flash flood caused by heavy rainfall and thunderstorms resulted in the death of 22 people and the flooding of roughly 45.000 houses and businesses and 70000 vehicles in Texas, Louisiana [13]. Similarly, in 2013 a flash flood in southern Ontario, Canada, incurred more than 850 million dollars worth of property damage [14].

The implementation of operational flood early warning systems becomes crucial. Thus, efforts have been made to produce hydrological predictive models capable of generating reliable and timely forecasts. These models can be divided into two main types: process-based and data-driven models. Process-based models use physical characteristics and physics laws to simulate the hydrological processes in the affected area. On the other hand, data-driven models predict flood occurrence by deriving insights from data. Data-driven models offered an efficient alternative to process-based models and delivered satisfactory performance in several hydrological applications, particularly flash floods [15, 16].

1.1 Problem and Motivation

There is ongoing research to enhance the predictive power of the data-driven flash flood models and extend the lead time of reliable forecasts. To predict the occurrence of floods, typically, past rainfall and streamflow magnitudes, along with other meteorological and hydrological variables, are used as part of the input features of the hydrological model. However, lately, it has been reported that integrating rainfall forecasts into process-based models improved their predictive performance [17, 18]. Similarly, utilizing precipitation predictions can help extend the performance of data-driven models. As a proof of concept, various studies introduced future observed rainfall values (which were considered perfect predictions) as input variables to data-driven models [19]. It was found that rainfall forecasts helped model floods effectively [20].

It is worth mentioning that rainfall predictions bring an additional source of uncertainty that can affect the model performance and thus should be studied and accounted for. As a matter of fact, it was found that rainfall predictions rep-

resent the primary source of uncertainty in process-based models [21]. However, studying the effect of rainfall forecasts on the data-driven models' performance has not been addressed yet. Therefore, this research focuses on quantifying the uncertainty of rainfall predictions and investigating their impact on the performance of a flash flood data-driven forecasting model. This research is built on top of the seminal work conducted by Erechtchoukova et al. [22], which proved the effectiveness of a data-driven flash flood prediction framework. The proposed framework used Machine Learning (ML) techniques to generate a model that predicts hydrological events (flood, non-flood) based on only data which are readily available, namely, rainfall and water level values generated by routine hydrological monitoring network.

1.2 Research aim and objectives

Studying the effect of rainfall forecasts on predictions generated by flash flood data-driven models has not been implemented previously. This thesis aims to investigate the impact of rainfall predictions on the performance of data-driven models while accounting for the uncertainty of rainfall forecasts. In order to attain this aim, we set the following objectives:

1. Review the existing literature on precipitation and flash flood prediction modeling and examine the related work to models uncertainty quantification.

2. Conduct a study to find and build an efficient model for very short-term rainfall forecasting using data solely from rain gauges.

3. Build a flash flood prediction framework.

4. Study the impact of generated rainfall forecasts on the performance of the data-driven models constructed under the flash flood prediction framework.

1.3 Thesis organization

This manuscript is structured as follows. Chapter 2 presents the theoretical background and the research related to flood modeling, rainfall modeling, uncertainty quantification, and the extracted research questions. Chapter 3 details the methodology adopted, the software tools, and the dataset used in this study. Chapter 4 describes the work conducted to build the rainfall prediction model and discusses the obtained results. Chapter 5 presents an application of the flash flood framework to a real case study, including quantification of the uncertainty of the rainfall forecasting model and comparative analysis of the effect of precipitation prediction uncertainty on the model's performance. Chapter 6 provides the conclusions generated from this study.

Chapter 2

Theoretical background, related work, and Research Questions

2.1 Theoretical background

2.1.1 Machine learning and Deep learning

Machine Learning (ML)

Artificial Intelligence (AI) is a sub-field of computer science that deals with automating tasks usually performed by human beings, such as cognitive tasks (perception, speaking, *etc*.). ML is one of the branches of AI that is concerned with extracting knowledge from data. During the last decades, the ML domain gained immense popularity and had notable advancements thanks to the increased capacity of computational resources and the availability of enormous amounts of data in electronic form. ML consists of a set of algorithms that 'learn' a set of rules and/or patterns from data (past experience) by optimizing a specific objective function [23].

Overall, there are three major branches of ML, namely supervised learning, unsupervised learning, and reinforcement learning. The data are composed of a set of input variables (called independent, explanatory, or features) and output variables, also referred to as target or dependent variables. In supervised learning, the values of the target variables are known. The goal of the ML algorithm is to find a good approximation of the actual function between the features and the output. On the other hand, in unsupervised learning, the target is unknown (it is also said 'the data are unlabeled'), and the objective of the ML algorithm is to discover unknown patterns or structures in data. Unsupervised learning includes tasks such as clustering, which is the organization of data into groups based on specific similarity measures. Other tasks involve extracting association rules (discovering relations between variables) and dimensionality reduction. Finally, in reinforcement learning, based on data gathered from the environment, the algorithm tries to find a sequence of actions that will maximize its reward or minimize its risk.

Supervised ML algorithms could be grouped into regression and classification algorithms. In regression ML problems, the target variable is continuous such as precipitation magnitudes or stock values. On the other hand, in classification ML problems, the dependent variable is categorical (nominal or ordinal). Depending on the number of categories, classification problems could further be classified into binary classifications where the output variable has only two values (e.g., flood or non-flood) or multi-class classifications where the target variable has more than two values (e.g., red, blue, orange, *etc*.). Based on the number of labels assigned to each input tuple, classification problems could also be categorized into single label or multi-label classification problems. In the multi-label classification problems, multiple labels could be assigned to each

tuple. To assess the performance of the supervised ML models, the data is usually split into a training set and a testing set. The ML algorithm uses the training set to learn the underlying function between the input and output variables. The testing set is used to evaluate the derived model performance on previously unseen data, i.e., to estimate the generalization error.

There are many regression and classification ML algorithms in the literature ranging from simple Linear Regression (LR) or Logistic Regression (LogR) models to Artificial Neural Network (ANN) models of various sophisticated configurations. ANNs, in particular, gained special attention and are extensively and successfully applied in many real-world problems [24].

ANN algorithms approximate the mapping function between the input and output variables through a set of units. These units were inspired by a biological neuron, which constitutes the primary information-processing cell in the brain. Based on a preliminary understanding, the brain neuron takes a set of incident signals from other neurons as input. If the linear combination of these inputs surpasses a certain threshold, the neuron fires (i.e., outputs an electrical signal).

Artificial neurons represent a simplified analogy to their biological counterparts. Each neuron computes a linear combination of incoming values, adds a bias, and applies a function to determine the output information. The applied function is called the activation or transfer function. Commonly used transfer functions include the Threshold, Sigmoid, Tanh, Relu, *etc*. Figure 1 depicts a simple conceptual representation of an artificial neuron.

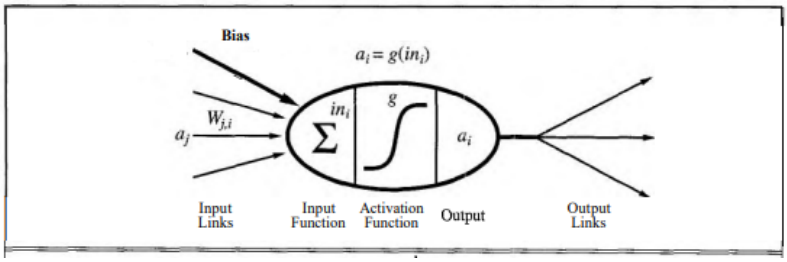

Figure 1: An artificial neuron 'i' computes a linear combination of its inputs and adds a bias. The neuron then applies an activation function to compute the output a_i [1].

Depending on how the neurons are interconnected, two main categories of ANN exist, namely, Feed Forward Neural Network (FFNN) (acyclic) and Recurrent Neural Network (RNN) (cyclic). In FFNN, the signal travels in one direction from the input to the output without any loops. In RNN, the output of a neuron is fed again as its input or as an input to previous neurons. Figure 2 illustrates the difference between FFNN and RNN. In the presented figure, The FFNN consists of one input layer, one hidden layer, and one output layer. The input layer does not perform any computations. It merely connects the input variables to the computational layers of the ANN. The hidden layer consists of the set of intermediate computational neurons which give their signal to the final layer that delivers the output. In the RNN model, output from the hidden units is fed again as its input and/or as input to the adjacent neurons in the same layer.

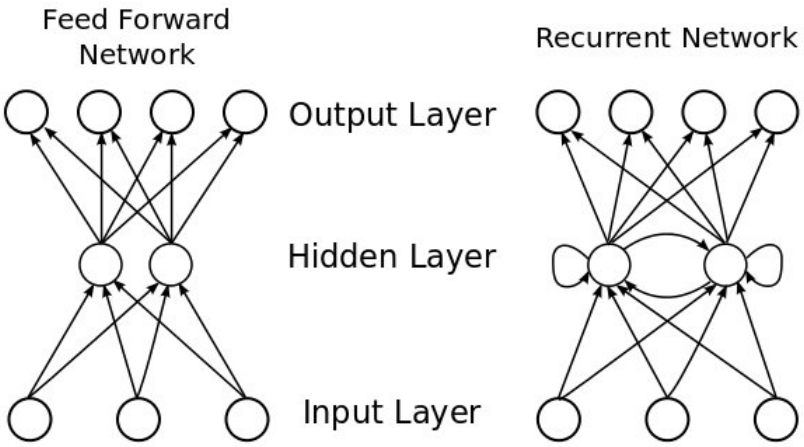

Figure 2: Difference between three layers FFNN and RNN [2].

During the training procedure, the ANN parameters are updated to minimize an error function. This function is known as the objective, the loss, or the cost function. Gradient Descent (GD) is one of the most used methodologies to optimize the loss function. Back-propagation (BP) is a training algorithm that applies GD to minimize the objective function. First, the BP algorithm computes the derivative of the loss function with respect to all the ANN parameters' using the chain rule. The parameters are then updated in the opposite direction of the gradient (taking a closer step toward the minimum). Figure 3 illustrates the steps of training an ANN [3]. Once the weights and the biases of the ANN are learned, they could be used to predict the output of previously unseen inputs.

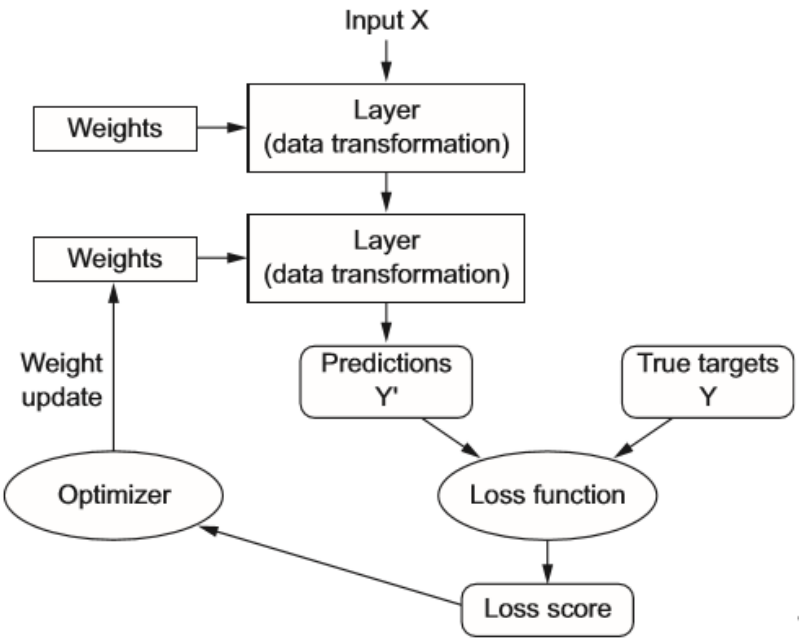

Figure 3: ANN training steps [3].

Deep Learning (DL)

DL is a subfield of ML that encapsulates various advanced configurations of ANNs. The distinguishing characteristic of these ANN models is their depth. DL models are composed of several layers (usually more than three layers); hence the denomination deep was given [3]. DL models established state-of-the-art performance in many tasks that were hardly achievable by traditional ML models. These tasks include image classification, speech recognition, machine translation, *etc*. It is interesting to mention that DL models were even able to exceed human-level performance in some of these applications. Thanks to the outstanding results and substantial amounts of research and applications achieved, the DL domain stood out as a subfield of its own. The main two areas of application of DL models are, first, the computer vision domain (e.g., image

classification and tagging), where Convolutional Neural Network (CNN) model is primarily used. CNN is a special type of ANN that consists of a set of layers (convolutional layer, Pooling layer, Feed Forward layer). These layers are used to convolute the input image with various filters and apply additional operations to extract spatial dependencies and patterns in the image data. The second area is sequence learning which includes tasks such as machine translation and speech recognition. RNN and its variants are primarily used in the latter area.

2.1.2 Uncertainty quantification in ML and DL

DL models are increasingly used in critical decision-making applications such as autonomous vehicles and medical diagnosis. Committing prediction errors in these risky situations can lead to disastrous repercussions, such as the fatality incurred by an assisted driving system erroneously labeling the sidewalk as the sky in 2016 [25]. This is where uncertainty quantification comes into play as it allows to specify the model's confidence in its predictions. Thus, when the uncertainty is high (based on a pre-determined uncertainty measure threshold), the control can be differed to the user to make the appropriate judgment. Uncertainty quantification allows as well to have a better understanding and interpretation of the model's performance and the variability of its predictions. Due to its significant importance, modeling and estimating the uncertainty in DL models has recently been extensively investigated [26, 27].

The overall uncertainty consists of two main uncertainty components: epistemic (derived from Epistemology, a field of philosophy dealing with knowledge) and aleatoric uncertainty. As its name might suggest, epistemic uncertainty comes from the model's limited ability to describe real processes due to its lack of knowledge (of the data). The epistemic uncertainty is also referred

to as the model uncertainty or the reducible uncertainty, as it can be reduced by collecting more data and improving the model structure. On the other hand, the aleatoric uncertainty is due to the inherent randomness in the data and the stochastic nature of the relationship between the input and output variables. The aleatoric uncertainty is irreducible and can be one of two types, namely, homoscedastic or heteroscedastic. The aleatoric uncertainty is homoscedastic if it is constant and does not depend on the input values; oppositely, it is heteroscedastic if it changes depending on specific input ranges. Figure 4 illustrates a schematic view of the aforementioned uncertainties in the case of a linear regression model.

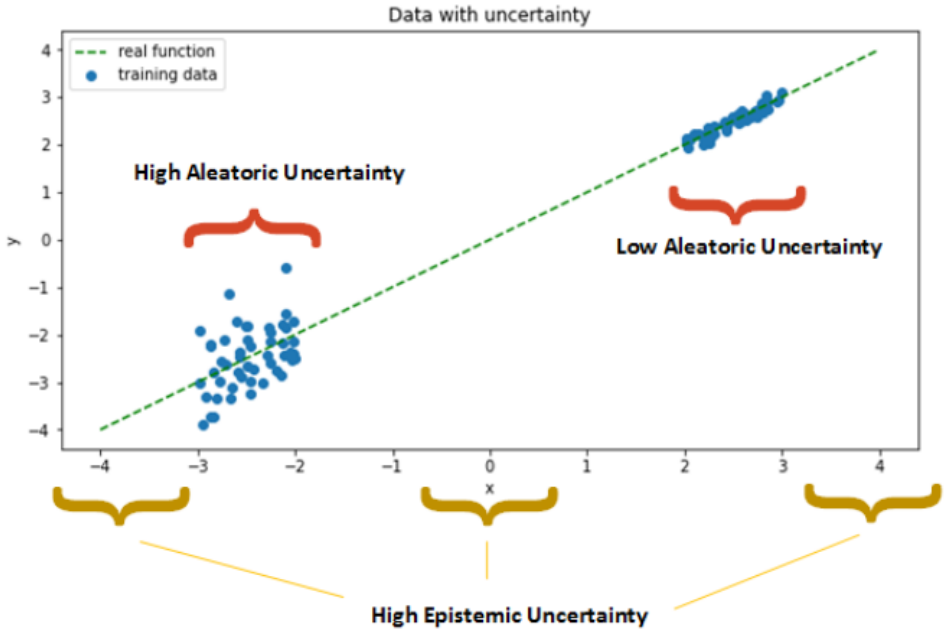

Figure 4: Uncertainty components illustrated in a linear regression model [4]

Across the literature, various techniques were proposed to model and quantify the epistemic and aleatoric uncertainties of DL models. While some studies focused on modeling each type of uncertainty separately, others tried to provide

general frameworks that account for the total amount of uncertainty. We will first present the most prevalent techniques used to quantify each uncertainty type. After that, we will introduce a general framework that accounts for the total uncertainty [28, 29].

Epistemic uncertainty

Based on the philosophical view of uncertainty, there are two approaches for epistemic uncertainty quantification in DL: probabilistic and non-probabilistic. Bayesian Neural Networks (BNN) and their approximations are among the most prominent uncertainty quantification methods based on Bayesian probabilistic theory. On the other hand, deep ensembles represent a non-probabilistic approach to uncertainty quantification.

BNN: As opposed to traditional ANNs, which learn deterministic parameters and provide point predictions using the Maximum Likelihood Estimation (MLE) principle, BNNs learn weights distributions and provide predictive intervals using Bayesian inference. Let θ denote the set of the ANN parameters'. BNNs are constructed by placing a prior distribution (which describes our prior beliefs) over the network parameters θ. A likelihood function is selected according to the prediction task. In the case of regression, the Gaussian likelihood is usually used. The likelihood determines the probability of getting a particular output given a certain input and the model's parameters. The posterior distribution is obtained using the Bayes theorem:

$$P(\theta \mid x, y) = \frac{P(y \mid x, \theta) P(\theta)}{P(y \mid x)} \tag{2.1}$$

where (x,y) represent input-output pairs of the training dataset D, $P(y \mid x, \theta)$

corresponds to the likelihood, $P(\theta)$ represents the prior distribution, and $P(y \mid x) = \int P(y \mid x, \theta) P(\theta)\, d\theta$ is referred to as the evidence.

Given a new input data x^* the predictive distribution is then computed by marginalizing over the posterior :

$$P(y^* \mid x^*, x, y) = \int P(y^* \mid x^*, \theta) P(\theta \mid x, y) \quad d\theta \tag{2.2}$$

However, due to the high dimensionality of the ANN parameters and for most of the prior, posterior pairs, the denominator (the normalization factor or the evidence) in equation 2.1 is analytically intractable [27].

A plethora of techniques was proposed to estimate the posterior distribution. These techniques can be classified into two main categories, namely, sampling and approximation methods. Sampling techniques consist of drawing multiple samples to estimate the posterior distribution. Monte Carlo Markov Chain (MCMC), Hamiltonian Monte Carlo, and Stochastic Gradient Langevin Dynamics (SGLD) are among the most used sampling methods. On the other hand, the approximation techniques are based on finding a surrogate distribution to the posterior that can be simply computed and marginalized over. The approximation techniques consist of two major methods: Laplace approximation and Variational inference (VI).

Laplace approximation consists of estimating the posterior distribution with a Gaussian. In comparison, VI methods consist of selecting a distribution from a family of variational distributions, which minimizes the distance to the posterior. The distance is measured using the Kullback–Leibler (KL) divergence measure, which is a measure that computes the distance between distributions. The diagram in Figure 5 illustrates a summary of the epistemic uncertainty quantification methods, which are based on BNN approximation methods.

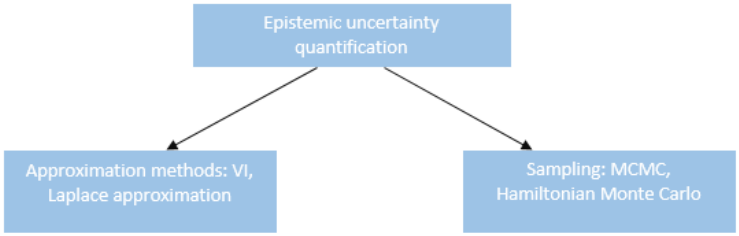

Figure 5: BNN approximation methods for epistemic uncertainty quantification.

The techniques mentioned above were found successful in many applications. However, they suffer from high computational costs [26]. In an attempt to solve this issue, Monte Carlo (MC) dropout was proposed [25, 30].

MC dropout MC Dropout is a regularization technique used to prevent the overfitting of the ANN model during training (overfitting occurs when the ANN fits the training data very well and performs poorly on the testing set, i.e., has a poor generalization ability) [5]. This technique consists of dropping (switching off) some neurons in each layer according to a certain probability p during each forward pass of the training process (Figure 6). The intuitive idea behind this technique is that a neuron in a subsequent layer can not rely on a specific neuron in the precedent layer and thus has to spread its weight across different neurons. Usually, this technique is used solely during training. However, in [31], the authors repurposed the use of this technique during inference and showed that applying dropout multiple times during inference corresponds to a VI approximation of a BNN posterior. Detailed proof of this technique can be found in [30, 31]. The predictive distribution can be represented by the set of outputs obtained from multiple passes through the network for each input value. The recommended number of forward passes usually ranges between 30 and

100 [32].

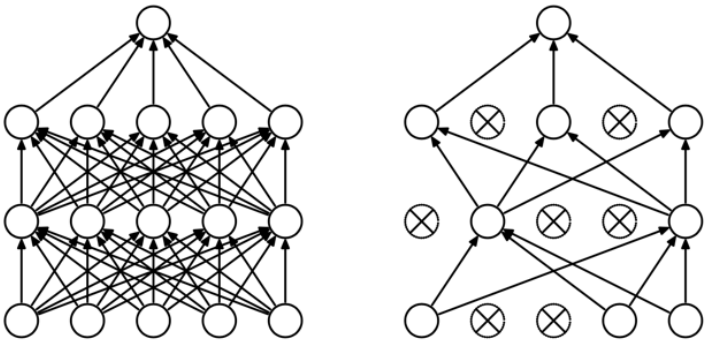

Figure 6: The dropout technique [5].

Deep ensembles Deep ensembles are among the most famous non-Bayesian approaches used to quantify the model epistemic uncertainty. These models are based on training multiple ANN models randomly initialized (by using different seeds). The variance of the model outputs represents the model predictive uncertainty [33].

[28, 29] presented a detailed literature review of existing methods for uncertainty quantification in DL and their applications. It was stated that MC dropout [30, 31] and ensembles [33] are two of the most widely used techniques for uncertainty quantification. These methods have been applied in a plethora of regression and classification applications. Among these application domains, there is computer vision (e.g., medical image diagnosis), natural language processing (e.g., text classification), *etc*.

In [34], the authors proposed an encoder-decoder model for daily uber rides time series prediction. The model epistemic uncertainty was quantified through the MC dropout method. [35] compared two techniques, namely, MC dropout and ensembles, to quantify the uncertainty of an LSTM-based hydrological model. The model predicted daily streamflow (which is the volumetric dis-

charge of water expressed in volume per time unit) based on rainfall and evapotranspiration. The researchers evaluated the performance of the uncertainty quantification methods through metrics such as the percentage of coverage that assesses the reliability of the predictive intervals and the average width that assesses their sharpness. It was found that the two uncertainty frameworks provided comparable results. No method consistently outperformed the other on all the uncertainty estimation performance metrics. Nevertheless, as MC dropout represented a considerable advantage in computational efficiency, it was recommended. Similarly, using the MC dropout method, [36] quantified the epistemic uncertainty of an Encoder-Decoder model developed for weekly precipitation forecasting.

Based on the various reviewed literature, it was stated that the MC dropout technique provided efficient model uncertainty estimates. This technique will be used to quantify the epistemic uncertainty of the DL models implemented in this endeavor.

Aleatoric uncertainty Aleatoric uncertainty is modeled by presenting the per-point estimate as a conditional Gaussian distribution with mean μ and variance σ^2. Adding another neuron to the output layer of a DL model is one of the well-known methods used to quantify the aleatoric uncertainty. This neuron will be trained to estimate the output variance due to the inherent data stochasticity. So the last layer of the DL model consists of two neurons: the first one estimates the mean, and the second one corresponds to the variance. If the modeled uncertainty is heteroscedastic, the output neuron corresponding to the variance depends on the previous layers. In the opposite case, the learned variance is constant. Figure 7 further explains how to transform a DL model to account for

the aleatoric uncertainty. Several other methods exist in the literature, including quantile regression [37].

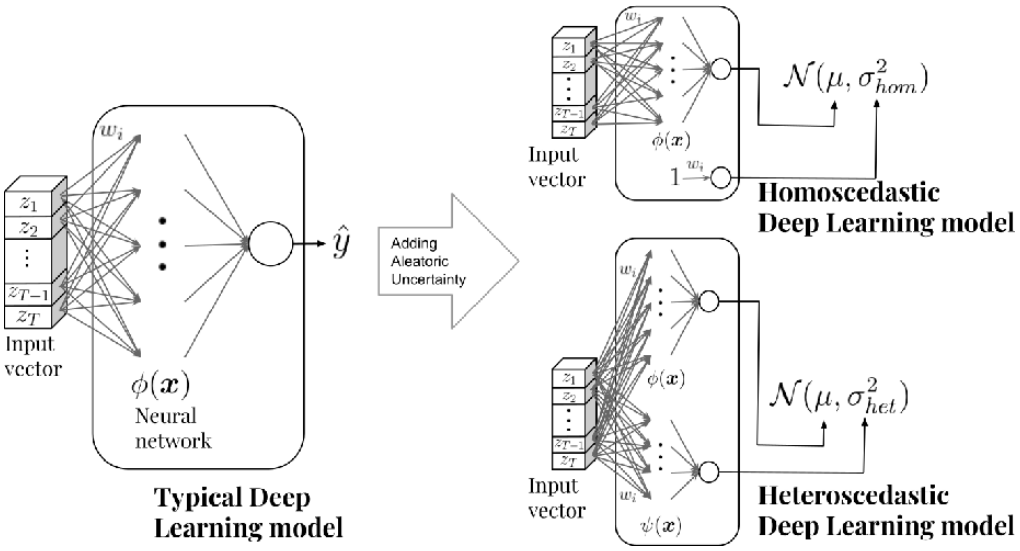

Figure 7: Modeling aleatoric uncertainty [6].

General framework for uncertainty quantification One of the recommended methods for overall uncertainty quantification is the coupling of the MC dropout (to quantify epistemic uncertainty) and the addition of another neuron to the output layer (to quantify aleatoric uncertainty). The overall variance of the output corresponds to the sum of variances due to the aleatoric and epistemic uncertainties. A detailed description of this framework is provided in [25].

2.1.3 Hydrological concepts

Hydrology is the field of science that studies the interaction of water and other environments. It could also be strictly defined as the study of the hydrological cycle. The hydrological cycle consists of the processes the water goes through from the atmosphere to the earth and back [7]. As shown in the illustrative Figure 8, these processes include precipitation, evaporation, and evapotranspi-

ration. The hydrological science finds its application in many problems such as hydropower generation, water supply, wastewater treatment, irrigation, *etc*. Particularly, predicting hydrodynamic characteristics of water bodies is essential for many problems, from reservoir management to water allocation to mitigating extreme hydrological events, among which floods are the most devastating. Flood management and warning domains have paramount importance as they deal with studying the flood formation processes and mitigating risks of the flood events. Modeling techniques vary according to many factors such as flood types, types of water bodies, and the hydrological and topographical characteristics of the watersheds (a portion of land wherein the water accumulates and drains to the lowest point). This section will present different types of floods and hydrological models.

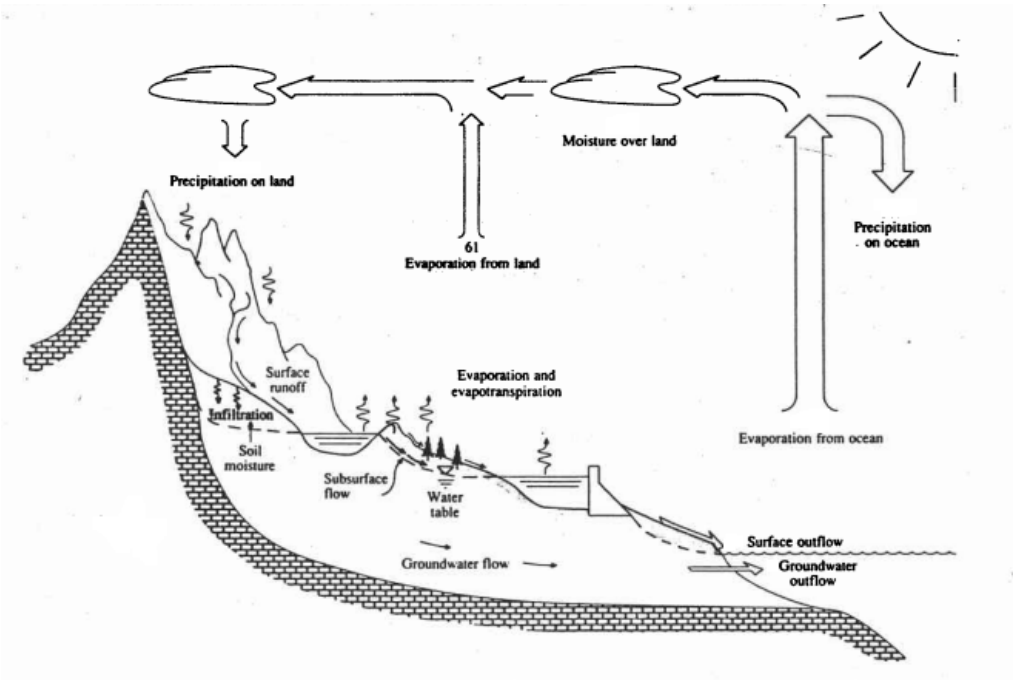

Figure 8: Hydrological cycle [7]

Flood types

A flood is any high-flow water that inundates areas that are usually dry. Various attempts were made to classify floods based on their deriving factors, duration, consequences, *etc*. Among these types, there is coastal flood (occurs when seawater overflows coastal land area due to underwater earthquakes or storm surges, *etc*.), urban floods (occurs when rainfall amounts exceed the capacity of drainage systems), *etc*. The two most common flood types incurred by precipitations are flash floods and river floods [38, 39].

- Flash floods: occur due to localized short intensive rainfall and can develop within a few minutes to hours. These floods usually take place in small or mountain watersheds with short response times (the time it takes water to flow from the highest point to the lowest point in a watershed).
- River floods: occur gradually over a long period of time, ranging from days to months. They are seasonal and are induced by repeated rainfall events and snowmelt. They usually happen in large basins.

As opposed to river floods, natural flash floods pose a particular challenge as they are swift, leaving little or no time to issue warnings and take mitigation measures [13]. Furthermore, nowadays, due to ongoing urbanization, impervious surfaces have increased substantially. Consequently, rainfall not absorbed by the soil quickly turns into runoff, increasing the chances of flash flood occurrence. Flash flood prediction is thus critical and paramount.

Hydrological models

Various modeling techniques were developed for flood prediction. The developed models could be divided into two main categories; namely, flow routing

models and rainfall-runoff models [13]. Flow routing models describe the flow movement upstream to downstream by solving fluid motion equations such as Navier–Stokes equations or their approximations. On the other hand, rainfall-runoff models represent the relationship between rainfall events and the river flow or surface water runoff. The latter models could further be classified into process-based and data-driven models [40].

- Process-based models: apply the physical principles to derive mathematical expressions to simulate the hydrological processes using the laws of conservation of mass, momentum, and energy.

- Data-driven models (black-box models): make no or limited assumptions about the physical processes of floods. These models represent floods by applying various data analysis techniques and extracting patterns from time-series data. ML and DL-based models are part of data-driven models.

To better understand the difference between the aforementioned models, Table 2.1 presents an overall comparison.

Table 2.1: Comparison of hydrological models.

Models	**Data-driven**	**Process-based**
Structure based on	Data	Physics laws
Interpretability	Low explanatory power	Easily interpretable, however, they require in depth expertise
Calibration based on	Historical data	Historical data

Process-based models usually require parameter calibration, which is both labour and time-consuming [41]. Conversely, data-driven models are cost-effective and were found to offer satisfactory predictive performance in a variety of hydrological modeling applications [15,42]. Therefore, data-driven models gained

popularity in the hydrological realm, particularly for the task of flood modeling. Hence, in this thesis, we focus on the application of data-driven models to flash flood prediction. More elaboration on the suitability of these models will be presented in the literature review section.

2.2 Literature review, foundation work, and Research Questions

2.2.1 Flood modeling related work

The review of scientific literature uncovered that flood prediction is implemented mainly based on process-based models, data-driven models, or combinations of these two types of techniques. Recently, due to the expansion of ML and DL domains, a lot of attention shifted toward the application of data-driven models for flood simulation. However, while part of the hydrological modeling community advocates using data-driven models, others are still in favor of process-based models thanks to their interpretability and transparency.

Various studies were conducted to compare the performances of these two types of models. [43] performed a comparative analysis between a hydraulic-hydrology model (Storm Water Management Model) and a Support Vector Regression (SVR) model for rainfall-runoff modeling in an urban area. Input variables for the process-based model included rainfall magnitudes and a set of parameters describing physical morphological characteristics of a catchment, such as slope, surface roughness, depression storage, and infiltration coefficients. Only rainfall and flow observations were fed to the SVR model. It was reported that the models delivered comparable performances, with the SVR

model having a lower Root Mean Square Error (RMSE) and a higher R^2. However, it was found that SVR failed to provide satisfactory performance when tested on out-of-distribution sample data, i.e., the test data had discrepant characteristics compared to the training data. Thus, the researchers pinpointed that efforts should be focused on tackling this issue.

In particular, Long Short-Term Memory (LSTM) models were extensively compared to process-based models. [15] investigated the performance of LSTM and FFNN models for hourly rainfall-runoff modeling. Antecedent rainfall values and water discharge (also referred to as streamflow) were used as input to predict the flow for the next one to six hours. The data-driven models were compared with two other process-based models. It was found that the black box models outperformed the other two models. Moreover, it was shown that LSTM delivered a higher predictive performance than FFNN, thanks to its memory cells' gating structure.

Likewise, [16] investigated the performance of LSTM for daily discharge prediction in 241 US catchments. The model was compared to a baseline process-based model (the Sacramento Soil Moisture Accounting Model (SAC-SMA) coupled with the Snow-17 snow routine). The LSTM model input variables included various meteorological characteristics such as observed minimum and maximum temperatures, vapor pressure, *etc*. Overall, three sets of experiments were conducted. The first set corresponded to training an LSTM model for each catchment separately. The second approach consisted of creating a common regional LSTM model for all basins. Finally, the third set of experiments corresponded to fine-tuning the regional models specifically for each catchment, i.e., the models trained on regional data are further trained using the data specific to each catchment (transfer learning). It was stated that using the third approach,

the LSTM model delivered comparable performance to the process-based model and was even able to slightly outperform it in certain basins.

Similarly, using the LSTM model, [44] proposed a framework for Spatio-temporal flood prediction. The LSTM model was integrated with the Reduced Order Model, which is used to reduce the input data dimensionality. The proposed model was then compared with a simulation model (a process-based model). The authors reported that the proposed data-driven model delivered comparable results to the process-based while reducing computational resources (CPU cost) up to the order of three.

In another study, [45] compared the performance of three data-driven models and a process-based model entitled SOBEK for the task of hourly streamflow prediction in two rivers with distinct hydrological characteristics. The data-driven models consisted of different variants of ANN, including the plain ANN, the LSTM model, and the Adaptive Neuro-Fuzzy Inference System (ANFIS) model. The first river consisted of a tidal river (41.61 km long with a basin area of 388.23 km^2), while the second was mountainous (has a length of 22.14 km with a basin area of 129.40 km^2). Hourly rainfall amounts, along with other meteorological variables, were used as input. The models were assessed in terms of model average accuracy, peak streamflow simulation, and model efficiency (validation and prediction times). Overall, it was found that the LSTM and SOBEK models were the best for average streamflow simulation, while ANFIS and SOBEK were the leads in terms of peak streamflow prediction accuracy. The ANN model, on the other hand, was the most efficient in terms of validation and prediction times. The use of the LSTM model was recommended as it compensates for the process-based model while having acceptable accuracy.

In an attempt to gain the best of both models types, other researchers tried to

integrate the performance of data-driven and process-based models. For example, [46] coupled two Support Vector Machine (SVM) models with the process-based MIKE model. One SVM model was used to predict flood occurrence, while the other was trained to estimate the flood depth. The authors used the MIKE model simulations to train and test the SVM models. It was revealed that SVM delivered high predictive performance on generated and real flood test data. The researchers suggested that further work should focus on determining an appropriate training sample size and taking hydraulic structure controls into account during the data-driven models' training phase.

[47] integrated the LSTM model with a process-based model for multi-lead times flood prediction. In the proposed approach, the discharge values predicted by the process-based model were fed to the LSTM model to predict true discharge values. Other variables, including observed and predicted rainfall amounts and past discharge values, were also used as input features. The forecast lead times ranged from three to 12 hours. Furthermore, The LSTM and process-based models were used separately as benchmarks. It was reported that the standalone LSTM model outperformed the process-based model for short lead times ranging from three to six hours. The latter gave better results than the data-driven model for longer lead times. It was also found that the hybrid model had a higher predictive ability than the other two models for all lead times.

[48] suggested an LSTM-based model for streamflow simulations post-processing for lead times ranging from one to seven days. The simulations were produced by process-based models. The constructed model added a self-attention mechanism to the LSTM cell. Compared to a set of ML and DL models, including Extreme Gradient Boosting (XGBoost), Gated Recurrent Unit (GRU), and LSTM, and to other operational models, it was found that the sug-

gested model had the best performance. Performance improvement was salient, particularly in snow-driven basins.

In most of the analyzed studies, data-driven models either outperformed or had comparable performance to their process-based counterparts. Furthermore, The main key advantage of data-driven models is their efficiency in implementation and the time needed for training and prediction. Hence data-driven models have been widely utilized in hydrological studies. The applications vary according to the specifics of the modeled hydrological phenomena, data availability, the forecast horizon, the target variable, *etc*. ML and DL models represent the most applied data-driven hydrological models.

Flood modeling using ML models

ML models were applied for both classification and regression problems. For classification problems, the target variable describes the occurrence of floods such as 'flood' or 'non-flood'. Depending on the study, these labels could be assigned based on the water level's exceedance of a certain threshold. In other cases, the target variable describes the range of water level values such as high, medium, or low. In the case of regression problems, the target variable corresponds to the value of water level or water discharge.

In the context of classification problems, [49] investigated several models for flash flood risk assessment. Risk levels (low, moderate, and high) of flood occurrence were determined based on various input features, including precipitation, temperature, wind speed, *etc*. The compared machine learning models included SVM, k-Nearest Neighbors (KNN), and Linear Discriminant Analysis (LDA). It was reported that SVM had the highest predictive performance in terms of precision and recall.

[50] presented a method for flash flood events classification based on cyclone Global navigation satellite system (GNSS) signal data. The authors used an ensemble-based algorithm entitled The Random UnderSampling Boosted (RUSBoost) for grid cell classification. Each spatial cell was classified as flood or land. It was stated that the model delivered satisfactory performance. [51] conducted a comparative study for water level categories prediction. The investigated approaches include Multilayer perceptron (MLP), MLP coupled with Genetic Algorithm (GA) optimization, and MLP coupled with BAT optimization. The optimization algorithms were used to determine the best ANN architecture. Further, a new approach that combined both GA and BAT optimization algorithms was proposed. The results showed that the proposed model presented a significant performance improvement of roughly 10% in prediction accuracy. The researcher suggested that the feature selection process can further enhance the model's accuracy.

Likewise, [52] compared several ML models for urban pluvial flood prediction. The model takes as input ten variables of rainfall intensities at different temporal scales and classifies each input instance as flood or non-flood. The subspace Discriminant Analysis was found to outperform the other ML models. The authors also reported that the proposed model outperformed the conventional thresholding technique, whereby a flood event is detected if the cumulative rainfall amount exceeds a pre-specified threshold.

Instead of using classification models to predict flood occurrence, other researchers estimated the water levels magnitudes or streamflow values using regression ML algorithms. For example, [53] compared several models for river flow forecasting. The investigated models included the persistence model (naïve), the linear transfer function (the model is a linear combination of an-

tecedent streamflow and rainfall values), the trend model (a linear extrapolation of previous streamflow values), and the ANN models. Several forecasting approaches were examined. The first approach consisted of creating a separate model for each lead time. The second approach consisted of a model that predicts multiple outputs simultaneously. The last approach consisted of generating a set of forecasts sequentially, .i.e, each forecast is fed into the model to generate the subsequent prediction. It was discovered that the first approach delivered the best results. Further, the results revealed that the linear transfer function gave better predictive performance for short-term forecasting (lead time ranging from eight to 40 hours). Nonetheless, for long-term prediction, ANN was found superior.

[54] applied the SVR model for flash flood prediction in a small mountainous catchment. The prediction lead time ranged from one to three hours. Lagged values of rainfall and runoff were used as input (i.e., rainfall and runoff values recorded at various previous time steps). The lag time of each input variable was determined based on sensitivity analysis and the catchment response time. The model was evaluated based on the peak flow and the time to peak error. It was found that the model delivered satisfactory performance (20% above or below the observed values). It was also revealed that runoff lagged values dominated the predicted runoff peak value (i.e., had the most considerable effect on the runoff peak value error). Nonetheless, lagged rainfall values reduced the time to peak error. The authors suggested applying the model to other catchments with different hydrological and topographic characteristics.

[55] compared several regression ML models for streamflow prediction. Lagged values of rainfall and discharge were used as input to predict next-day streamflow. Input selection using Auto Correlation Function (ACF), Cross-

Correlation Function (CCF), and Partial Auto Correlation Function (PACF) was done to pick the appropriate lag steps of rainfall and discharge input variables. The compared models included Least Squares SVM (LSSVM), Multiple Linear Regression (MLR), and MLP. Two nature-inspired algorithms were applied for the models' hyperparameter optimization, namely Particle Swarm Optimization and Harris Hawks Optimisation. It was indicated that LSSVM coupled with Harris Hawks Optimisation delivered the best predictive performance.

Similarly, as an attempt to model discharge, [56] proposed a hybrid ML model. The established model consisted of a FFNN for discharge estimation and a KNN model for error estimation; the two values were then summed to provide the forecast. Antecedent rainfall and runoff values were used to predict the next hour's runoff. Further, GA was used to optimize the ANN topology, whereas Levenberg–Marquardt back-propagation was applied to learn the model parameter. It was stated that the proposed model gave a satisfactory performance.

[57] studied a set of machine learning models for short-term water level forecasting under heavy rainfall events. Historical rainfall and water levels values were used as input. Various scenarios were considered by varying the number of lags of input variables and training the models using heavy rainfall events solely or the whole data. The models' performance for different lead times was also tested. It was reported that ANN delivered better performance when trained on rainfall events instead of the whole dataset. Four-hour forecast based on the past two hours provided satisfactory performance for all models. It was also stated that ANFIS was more resistant to noisy data while SVM was more robust during heavy precipitations (no fluctuation).

Based on the presented literature, it can be seen that ANN, alongside SVM and SVR, were widely used for rainfall-runoff modeling. However, DL models,

particularly RNN and their variants, have been increasingly used for hydrological modeling. Hence, many studies were conducted to analyze the performance of these model types.

Flood modeling using DL models

[58] investigated RNN and MLP for rainfall-runoff modeling. The authors addressed several issues. The first one is the ANNs extrapolation problem which is defined as the incapability of ANNs to predict values beyond the range of their training data. It was stated that standardizing the input helped mitigate this problem. Second, the use of distributed and areal average rainfall values as input variables was studied. It was found that lumped rainfall values speed up training with no accuracy loss.

[59] studied the reliability of short-term multi-step models for Floodwater Storage Pond (FSP) prediction. The compared models included Elman RNN, Back Propagation Neural Network (BPNN), and Nonlinear Autoregressive Network with Exogenous inputs (NARX). The forecasting window ranged from 10 to 60 minutes. Two scenarios were investigated depending on the input features considered. In the first scenario, both rainfall and FSP levels were used as input variables, while only precipitations were considered in the second. It was found that, for scenario one, the recurrent architecture-based models (Elman RNN and NARX) outperformed the BPNN. The ELMAN model delivered the best performance for lead times ranging from 10 to 40 minutes, while the NARX gave better performance for higher lead times. For scenario two, the NARX model gave the best performance for all lead times. Furthermore, overall the authors found that the first scenario delivered better results than scenario two, emphasizing the importance of using FSP as an input variable.

RNN variants such as GRU and LSTMs were often used for hydrological simulations. For instance, [60] used GRU to predict the next 24 hours of stage values at the location of interest. The input features consisted of historical water levels at the target location and upstream gauges along with antecedent precipitation measurements. Two schemes were considered depending on the input features' number of lagged values. GRU was compared with the traditional FFNN. It was reported that GRU outperformed FFNN and delivered a high predictive performance for the studied watershed.

Further, [61] proposed a Spatio-temporal LSTM model for streamflow prediction. The established model was compared with a set of statistical and deep learning models such as Historical Average, traditional LSTM, Graph Convolutional Networks, *etc*. The authors reported that adding the attention mechanism considerably improved the model performance. It was also ascertained that the attention weights changed gradually according to the forecast time, which could have physical interpretations based on the basins' response time.

Another variant of the LSTM model, Cyclic LSTM (LSTMC), was applied in [62] for long and short-term flood prediction. The observed rainfall values along with reservoir water levels were used as input. The flow was predicted for the next 300 hours. Data cleaning and prepossessing included missing values imputation using the Inverse Distance Weighting method. The mutual information analysis was done to select the appropriate time delay for the input variables. The researchers compared the performance of the proposed model with LR and BPNN. It was reported that the proposed model outperformed its counterpart and delivered an acceptable performance in terms of flood peak arrival time and value (within the permissible error range according to the standard for hydrological information and hydrological forecasting).

[63] proved the adequacy of LSTM for streamflow forecasting. A set of experiments was conducted to evaluate the effect of the training set size, the time difference between the training and testing set, and the prediction lead time on the predictive performance of the LSTM model. [42] showed the suitability of an LSTM model for water level prediction at five urban river sections. In this study, only antecedent water levels were fed to the model. The authors proposed using the rolling forecast methodology, where a generated forecast is updated according to a rolling interval. It was revealed that for two hours prediction period and 30 minutes rolling interval, the LSTM model efficiently predicted water levels.

LSTM showed higher performance in a set of other comparative studies. [64] compared the performance of a group of (traditional) ML and DL models for daily streamflow forecasting. The investigated models included LSTM, LR, MLP, and SVM. The input features included historical rainfall and streamflow values. Several cases were evaluated based on the number of lags of the input variables. It was reported that LSTM showed the best performance in predicting streamflow in the studied river basin. Moreover, the author found that the last two days' precipitation and streamflow values delivered better performance than other combinations.

[65] conducted a comparative study of LSTM, SVR, and MLP models for real-time water stage forecasting. Prediction lead times varied from one to nine hours. Input features consisted solely of observed water levels upstream and downstream from the target location. Downstream water levels were taken into account to consider backwater. The number of lag steps on input features was determined based on a correlational analysis. All the studied models delivered accurate predictions within the acceptable error range for forecast horizons rang-

ing from one to six hours. However, the performance deteriorated significantly for higher lead times. Furthermore, it was reported that out of the three compared models, MLP had the largest RMSE while LSTM and SVR had comparable results, with SVR performing slightly better. Nonetheless, it was found that LSTM significantly outperformed the others in predicting peak values.

Integrating future rainfall values for data-driven flood modeling

Hitherto, in the presented data-driven hydrological models, input variables corresponded to past rainfall, water levels or streamflow values, and other hydrological and meteorological variables. However, it is interesting to mention that introducing rainfall forecasts as input features were found to substantially improve process-based models' forecasting ability [66].

[67] studied the impact of integrating precipitation forecasts into a process-based hydrological model for short-term flood prediction. Two types of rainfall forecasts were considered: numerical rainfall prediction and ensemble precipitation forecast. The authors stated that coupling rainfall prediction with the hydrological model significantly improved flood predictions with a decrease in relative error reaching 40% as opposed to not introducing rainfall forecasts. Particularly, it was reported that rainfall ensemble forecasts had a higher impact on the hydrological model performance. [18] ascertained the importance of using rainfall forecasts as part of the input variables to a process-based model. It was also found that enhancing short-term rainfall forecasts accuracy substantially improved the investigated models' performance.

It is interesting to mention that, from a practical point of view, flood management authorities usually use rainfall forecasts to guide their decision-making process about flood occurrence. Therefore, lately, the introduction of rainfall

forecasts into data-driven hydrological modeling has caught the attention of several researchers.

[20] applied several ensemble models for short-term flash flood predictions. The studied models included XGBoost, Random Forest, and Rotation Forest. Two scenarios were investigated according to the models' input features. In the first scenario, past observed rainfall and water level values were used as input. However, in the second scenario, future rainfall values were also considered. Experimental studies revealed that XGBoost delivered the best performance for lead times spanning 15 minutes to 60 minutes. Moreover, introducing future rainfall magnitudes significantly enhanced the models' predictive skill for lead times higher than 90 minutes.

In [68] study, a sequence to sequence LSTM model for hourly rainfall-runoff modeling was applied. Prediction lead time spanned 24 hours. Observed and predicted rainfall values, past runoff values, and monthly evapotranspiration measurements were used as input features. Several data preprocessing techniques were applied to optimize the model performance. First, through trial and error, the time delay for the input features was set to 72 hours. Second, the Moving Averages (MA) technique was used to smooth the rainfall data. It was found that MA decreased the noise significantly and improved the model's predictive ability. Hyperparameter tuning was also conducted to optimize the model performance. It was reported that the model significantly outperformed its counterparts which included the persistence model, MLR, SVM, Gaussian Process Regression, and regular LSTM. Moreover, it was stressed that the model is watershed specific. Therefore, model calibration must be conducted for each watershed separately.

In a subsequent work [69], the authors applied a GRU-Based RNN model for

runoff prediction in downstream gauges. In this study, observed and predicted rainfall were used as input. Further, past flow values in upstream and downstream areas were also included in the input feature set. Predicted streamflow in the upstream area was added to the input features set to predict streamflow downstream. The lead times ranged from one to 120 hours. It was established that the proposed model significantly outperformed Persistence, Ridge Regression, and Random Forest regression on the majority of the gauges. The proposed model outperformed the LSTM sequence to sequence model [68] for lead times exceeding 24 hours.

Similarly, [19] proposed a discharge forecasting model based on the LSTM model for flash flood forecasting in mountainous areas. The input variables included past discharge values, observed rainfall quantities, and future precipitation values. The lead times ranged from one to 10 hours. The proposed models delivered satisfactory performance. In a follow-up work [70], the authors applied the same model for the discharge prediction problem. However, they studied the uncertainties incurred from the sampling approach, the modeling approach, and the model architecture. The authors emphasized the importance of quantifying these uncertainties and their impact on the model performance.

These studies proved the usefulness of integrating rainfall forecasts into data-driven hydrological models. However, as proof of concept, future observed rainfall amounts were used as perfect predictions. Nevertheless, [71] considered two scenarios. Ground truth rainfall values were used as perfect forecasts in the first case. In the second case, rainfall predictions were simulated by adding Gaussian noise with mean zero and varying standard deviation corresponding to the prediction lead times. The rainfall values and other hydrological variables (e.g., runoff) were fed to a proposed LSTM-based sequence-to-sequence model

to predict runoff for the subsequent seven days. The authors compared the performance of their model to a set of other sequence-to-sequence models using both scenarios. It was reported that, in general, their model outperformed the other LSTM-S2S models on the set of basins considered for their study.

Limitations To conclude, even though some studies introduced future observed precipitation values into data-driven flood prediction models. Assessing the impact of rainfall forecast uncertainties was not fully addressed. Particularly, the effect of rainfall predictions was not previously studied for short-term flash flood prediction models (where precipitation forecasts constitute the primary source of uncertainty).

2.2.2 Rainfall modeling related work

Precipitation is a fundamental element of the hydrologic cycle. It plays a vital role in all aspects of human lives. Therefore, over decades researchers made major efforts to understand, analyze, estimate, and predict this complex phenomenon. The most widely used techniques for rainfall observations include rain gauges, weather radar, and satellites (both geostationary GEO and low earth orbiting LEO) [13]. Based on the data generated from these monitoring systems, a plethora of rainfall prediction methods were developed to forecast precipitation at different temporal and spatial scales.

The World Meteorological Organization [72] defines various types of forecasts according to the prediction lead time as follows: nowcasting (0-2h), very short-range (2-12h), short-range (12-72h), medium-range (72-240h), extended-range (10-30 days), and long-range (30 days to two years) forecasts. The lead time of the predictive model corresponds to the scale of the meteorological

phenomenon that resulted in the rainfall event. The scales of the meteorological phenomena include microscale (50-2 km), mesoscale (2km-20km) and synoptic-scale (20km-200km) [73]. Rainfall nowcasting models, in particular, are critical. They help improve the predictability of rainfall-triggered natural disasters such as flash floods. In this section, we will present a general overview of the techniques used for rainfall modeling, emphasizing rainfall nowcasting methods.

Precipitation is a stochastic process; thus, many models were proposed to enhance its predictability. Similar to the hydrological models, rainfall prediction models can be divided into two major categories: process-based and data-driven models. Process-based models simulate the underlying physical process that generates rainfall. On the other hand, data-driven models make no prior assumption about the physical characteristics of the phenomenon being modeled. Insights about the model structure and parameters are inferred from the data.

The most acclaimed Numerical Weather Prediction (NWP) models are process-based. These models are commonly used for long- and medium-term forecasts. Their computational cost is very high, and it could take up to six hours to compute a forecast [74]. NWP models suffer from the uncertainty of initial and boundary conditions. Among other issues, NWP models suffer from the spin-up time requirement, which is the time needed by the model in order to adjust from the initial conditions to a physically valid state [75,76]. Lately, with the advances in computational resources, NWP models have been used for short-term rainfall prediction. However, they still suffer from multiple limitations [76]. First, in order to provide short-term forecasts, complicated data assimilation procedures must be conducted. Further, a large amount of data with high spatial and temporal resolution must be provided [77–81]. Data-driven models offer an

efficient alternative. This review, therefore, focuses on data-driven short-term rainfall forecasting techniques.

Along with the technological advances in computational resources, data-driven modeling for rainfall forecasting gained a lot of attention. The proposed models can be partitioned into three main subcategories, namely, radar echo extrapolation techniques, statistical modeling, and ML tools.

Echo Extrapolation rainfall modeling technique

Radar echo extrapolation methods predict rain by extrapolating rain cells' motion in radar maps. Various extrapolation schemes were described in the literature. [82] tried to forecast rainfall amounts using exponential, linear, and power-law extrapolations. The authors applied these techniques on radar maps to predict flux and area changes in rain cells during the subsequent three hours. It was found that linear extrapolation delivered the lowest prediction error; nonetheless, overall, compared to the status quo Persistence method, extrapolation schemes showed no significant improvement. [83] also used a linear extrapolation scheme and tried to quantify the error in rainfall forecasting using radar data. It was reported that for 30 minutes forecast, the error reached 50%, and for three hours forecast, there was a 60% error.

In an attempt to enhance performance, researchers tried to combine radar maps' extrapolation methods and NWP model output. For example, [84] carried out a study to characterize extreme rainfall events and assess the predictability of a short-term precipitation forecasting model by incorporating radar maps' extrapolations and NWP data. The authors reported that for a three hours lead time, only 20% of the events had an accumulated rainfall which was correctly estimated with an error of less than 25%. In [76], the authors proposed a model

that integrates NWP and radar-based extrapolation techniques. In the developed model, radar extrapolations were no longer based on the assumption of persistence, however predictions using the NWP were taken into account. Forecast performance for four hours lead time was not satisfactory. These results clearly show that radar echo extrapolation techniques need further enhancements.

Statistical and ML-based rainfall modeling

Statistical techniques have also been extensively used for rainfall forecasting. These models gained popularity due to their simplicity and easy implementation. They consist of a set of time series analysis techniques that decompose time series into constant, trend, and noise components. This decomposition can be done in various ways. Among statistical models, there are Autoregressive Integrated Moving Average (ARIMA), Multivariate Autoregressive Integrated Moving Average (MARIMA), Vector Autoregression (VAR), *etc*. In addition to statistical models, ML models were also extensively used to model precipitation. Many models were developed to forecast rainfall at different lead times. Due to the wide range of existing modeling approaches and the geographical and meteorological specifics of the modeled rainfall phenomenon, many researchers undertook an attempt to find the appropriate empirical model for the modeled phenomenon.

A survey of literature dating back to 1990 detailed the use of ANN of various configurations such as BPNN, Radial Basis Function Neural Network (RBFNN), multi-layer FFNN, Time Delay Neural Network (TDNN), for the task of rainfall prediction and compared it to other statistical models such as ARIMA and numerical models [85]. The prediction lead times ranged from hours to months and years, and multiple types of meteorological variables were used as input.

Overall, the authors noted that in the majority of the reviewed papers, ANN models were found superior in terms of predictive skill.

A set of studies were proposed to estimate future monthly rainfall amounts. [86] compared the Seasonal Autoregressive Integrated Moving Average (SARIMA) and Autoregressive Neural Network (ARNN) for monthly rainfall forecasts in Thailand. They used 146 years of monthly rainfall amounts to calibrate the models and reported that ANN delivered better performance compared to the SARIMA model. [44] established a new approach entitled Bayesian Enhanced Modified Combined Approach (BEMCA) to forecast monthly precipitation measurements. The proposed method consisted of updating the weights of an ANN model based on a Bayesian probabilistic framework. Several other variants and models were compared, such as ARMA-based models. Experimental results showed that the ANN-based approach delivered better performance. SVM model also showed significantly superior performance compared to the SARIMA model for monthly precipitation forecasts in a study conducted by [87].

[88] conducted an extensive study to find the best preprocessing techniques among Average Mutual Information (AMI), Linear Correlation Analysis (LCA), Partial Mutual Information (PMI), Principal Component Analysis (PCA), MA, Singular Spectrum Analysis (SSA), and the best-performing algorithms out of ANN, LR, Modular Artificial Neural Network (MANN), KNN for the rainfall prediction task. Methods were investigated for daily and monthly rainfall forecasts. The combination of the LCA feature selection method along with the SSA data preprocessing technique and MANN had the best predictive performance. However, the authors pinpointed that further effort needs to be done in forecasting peak values.

Conversely, [89] investigated the Persistence, the ANN, and the multivariate

time series analysis models ARIMA and VAR for rainfall nowcasting. Prediction lead times ranged from one to 12 hours. it was found that the VAR model gave the best result, followed by ARIMA. Unlike previous studies, ANN was outperformed by other models. Therefore, we cannot make prior assumptions about the appropriateness of a certain model for a certain case study. As stated by the no-free lunch theorem, that proves the nihility of a universal learner capable of learning all tasks ([90]). i.e., for each learner, there exists a task where it fails while another learner succeeds (achieves zero loss).

Similar to monthly forecasts, many ML models were proposed for daily forecasts. [91] compared ANN, Decision Tree (DT), KNN, and Rule learning system for daily precipitation prediction. Various meteorological variables were considered. A feature selection process was then applied (using the information gain metric and statistical tests) to retain the most relevant features. After preprocessing and hyperparameter tuning, experiments revealed that the rule-based classifier delivered the best performance.

In [92], rainfall amounts from previous days were used to predict the next day's precipitations. The compared models included RBFNN, MLP, Model Tree (MT), ANN, and Time Lagged Recurrent Neural Network (TLRNN). To assess models' performance, the authors used the Correlation Coefficient between observed and predicted values, the RMSE, and the Coefficient of Efficiency evaluation measures. MT and ANN were found to have comparable and remarkable performance as opposed to other algorithms.

As opposed to the previous regression models, [93] investigated a set of ML and statistical approaches for daily rainfall events classification. A set of meteorological factors such as temperature, humidity, wind speed, and direction were used as input. LogR was found to give higher predictive skill in terms of

recall, whereas DT outperformed all other models in terms of precision.

[94] proposed a model based on LogR for daily rainfall prediction and compared it with the state-of-the-art NWP model. Grided rainfall estimates obtained from satellite images combined with data obtained from rainfall gauges and radar were used as input. First, pixels from previous days' images that have a high correlation with the current state are used as input to the LogR model. The learned model then provided the probability of rainfall occurrence. The experiments demonstrated that the ML model outperformed climatology-based models in wet tropical areas, whereas the latter models showed higher performance in arid areas.

Statistical and ML-based models were also extensively applied for short-term rainfall prediction. [95] compared the performance of MLP and ARIMA models for very short-term orographic rainfall prediction (approximately four hours in advance). The authors used precipitation measurements recorded by a laser precipitation monitor as input. It was reported that ANN outperformed the ARIMA model even though the latter had acceptable performance. [96] compared the performance of several data-driven models for rainfall prediction. The prediction window ranged from one to six hours. Studied models included statistical (ARIMA and Autoregressive Moving Average (ARMA)) and ML models (ANN and KNN). It was reported that ANN delivered the best performance and improved rainfall run-off modeling.

[77] compared two models for intensive precipitation events classification. The first method used the statistical fingerprinting technique, which consists of detecting a pattern (a fingerprint) that characterizes extreme events. Events that show a similar pattern (analogous to the fingerprint) are classified as extreme. The second model consisted of applying the SVM model to a set of input fea-

tures that were selected based on an Anomaly frequency Method (AFM) (features that frequently portray anomalous behavior during extreme events were selected). Classifications were provided with a lead time of 6-48 h. Even though SVM provided a higher detection rate than the fingerprinting technique, both methods suffered from a high false-positive rate.

In another study, [97] used Precipitable Water Vapor (PWV) estimated based on data from the Global Navigation Satellite System (GNSS) to forecast rainfall amounts for a six-hour lead time. The authors found that PWV measurements increase significantly before the occurrence of precipitation. Based on this, monthly PWV, PWV variation, and PWV rate of change were used as input to fit a polynomial function. It was reported that the created model gave satisfactory results. However, it suffered from a high false alarm rate of 65%.

[98] tested several machine learning models for rainfall nowcasting. Models included ANN, Random Forest (RF), LR, and SVM, among others, and were trained on a set of meteorological variables. Variables that were highly correlated with rainfall amounts were retained. Among selected variables, there were relative humidity, atmospheric pressure, wind speed, *etc*. Datasets corresponding to regions with different geographical spread (one small region and one big region) were used for testing. The authors reported that RF and ANN delivered the best performance, with random forest outperforming ANN in the bigger region and vice versa.

[79] established a model for short-term intensive precipitation nowcasting ($>$20 mm/h) based on radar maps. The researchers integrated a graph model and a Random Forest (RF) model to deliver hourly forecasts of short-term intensive rainfall events. The graph model was used to represent convective cells. Extracted features are then fed into a RF model to classify the rainfall event as

intensive or not. RF model was then compared with two radar maps extrapolation techniques (Tracking a Radar Echo by Cross-Correlation (TREC) and optical flow method). It was reported that the proposed model significantly outperformed compared models.

Taking a closer look at the analyzed papers, we found that ANN was the most extensively used ML.

ANN for rainfall modeling

The ANN models were used to model rainfall at different lead times. [99] provided a detailed review of 43 papers using ANN for water resources modeling at different temporal resolutions spanning from years to minutes. The authors discussed the issues related to the application of ANN for hydrological modeling. Particularly, issues related to the modeling process were of primary concern. These problems included: the use of validation data for model evaluation, the choices related to the number of training iterations, and the network architecture (number of layers, number of nodes per layer, *etc*.). It was concluded that the ANN model development process should be systematized, and guidelines should be provided.

Likewise, [100] presented a survey on the use of ANN models for rainfall prediction. Among the studied models, there are FFNN, BPNN, TDNN, and RNN. Lead times ranged from weekly, monthly to yearly. Based on the reviewed papers, the authors found that in many case studies, ANN models outperformed statistical (ARIMA, VAR, *etc*.) and NWP models. Furthermore, researchers reported some major issues with the application of ANN to rainfall prediction, particularly peak values underestimation.

[101] presented a study for monthly rainfall prediction using ANN net-

works of several configurations: BPNN, RNN, and Continuous Basis Pursuit (CBP) models. Humidity, wind speed, and average rainfall were used as input variables. Based on the Mean Squared Error (MSE), BPNN was reported to be the best model for the precipitation prediction task.

Similarly, to model monthly rainfall predictions, [102] conducted a sensitivity analysis to select a combination of meteorological variables as input features. ANN and ANFIS models were trained on the selected features. ANN had a lower error and higher correlation coefficient than the ANFIS model. It was also reported that vapor pressure was the most important factor for rainfall forecasting.

Several other studies were conducted to predict monthly precipitation magnitudes [103, 104]. ANN models were also broadly used for short-term rainfall prediction tasks. In [105], the authors established a FFNN model using prior precipitation amounts and wind direction. The model provided six hours forecasts. It was reported that the model significantly outperformed the Persistence method in terms of the Correlation Coefficient, Skill Score and RMSE. However, it was pinpointed that other input variables are needed in order to enhance the model predictive ability.

Comparably, [106] compared several ANN architectures (different numbers of hidden nodes and types of activation functions) with the Persistence model for the task of rainfall prediction over the next one to six hours. It was found that ANN outperformed the Persistence model. Also, using the sensitivity analysis, the authors found that wet-bulb temperature was the second important input variable after observed rainfall amounts.

Based on the MLP model, [80] devised a new approach entitled Dynamic Regional Combined short-term rainfall Forecasting (DRCF) for three-hours rain-

fall prediction. The approach takes into account the occurrence of rainfall in neighbouring regions to make forecasts in the target area. Various high altitude and surface-level input variables were used as input features. PCA was then applied for dimensionality reduction. The proposed model showed higher performance in terms of RMSE and Threat Score (TS) compared to other physical (NWP), ML (RBFNN, SVM), and statistical models (ARIMA).

[107] integrated ANN with the multi-sensor data from rain gauges for short-term rainfall prediction (up to 150 minutes lead time). GA was used to select an informative subset of rain gauges. The authors applied sensitivity analysis to determine the best lag times for input variables and the best surrounding stations. It was found that ANN coupled with GA for network optimization consistently outperformed the standalone ANN model.

DL-based rainfall modeling:

Recently, with the emergence of the concept of DL, a lot of the attention shifted toward the application of DL models in the meteorological modeling domain. A lot of research was thus done to model precipitation using DL models.

[108] conducted a comparative study between statistical (Holt-Winters and ARIMA) and DL models (LSTM, Intensified LSTM, RNN) for rainfall prediction three months in advance. The authors found that DL models significantly outperformed statistical models in general. Particularly, intensified LSTM delivered the best performance and was able to tackle the exploding and vanishing gradient problem.

In another set of comparative studies between DL models and ML models, [109] investigated the performance of LSTM, RF, and XGBoost for precipitation prediction. Radar images were used as input, and precipitation estimation

for the next ten days was given. [110] compared five rainfall forecasting models using weekly rainfall averages as input. Compared models are LSTM, ANN, SVR, DT, and RF. Univariate time series data were used as input. The results showed that LSTM delivered by far the best performance. Similarly, using univariate rainfall time series data, [78] investigated the performance of ARIMA and LSTM models for daily rainfall prediction.

[111] compared Bidirectional LSTM (BiLSTM), Attention LSTM, and Attention BiLSTM for precipitation forecasting two days in advance. Several meteorological variables were used as input and underwent a set of prepossessing procedures (handling missing data, normalization, smoothing, *etc*.). Empirical results showed that BiLSTM outperformed other models. It is important to note that the authors pinpointed that the model needs further experimental validation using more data.

The experimental results in all of these studies showed that the LSTM algorithm notably outperformed its counterparts. However, one of the issues that were frequently raised by the authors is the inability of the LSTM model to accurately predict outliers and intensive values.

Deep Learning models were also broadly used for short-term forecasts ranging from minutes to hours. Particularly CNN was proved to have good predictive performance when used with satellite and radar products. [74] trained a CNN model called U-net on a combination of satellite and radar images along with ground station data. The model takes as input a sequence of images and predicts the next few hours' images. The established model was compared with the baseline Persistence model along with another NWP model (HRRR). It was reported that the U-net model significantly outperformed the others for short-range forecasting windows.

[112] proposed a rainfall prediction model and tested it for three hours and one day lead times. A multi-tasking CNN algorithm was used to train the model (multi-tasking refers to the ability of the model to predict rainfall at multiple locations). Model performance was compared to several other physical (ECMWF NWP) and data-driven models (LR, MLP, RNN). It was proven that the proposed model outperformed the other models.

The U-net CNN model was also trained by [113] to semantically segment satellite imagery into light, moderate, and heavy rain regions. Forecasts are provided for two hours with ten minutes updates. A combination of satellite images, radar, and NWP data was used as input. Results showed that the proposed model outperformed physics-based algorithms.

In another study, [114] made use of a CNN model to predict rainfall over the next hour or the next day. The model used IR (Infrared) and MW (MicroWave) data obtained from satellites. The model consisted of two modules, one of which is used to detect rainfall occurrence and the other to estimate precipitation measurement. The created model performed better than other satellite-based rainfall prediction models. However, it was noted that it underestimated peak values.

GRU models were also explored for the task of rainfall prediction. [115] compared several models including, XGBoost, MLP, and GRU, for rainfall prediction using various radar and weather station data. Compared to XGB and MLP algorithms, GRU delivered the best results for one, three, and six hours forecasts. [116] also leveraged GRU-configured deep learning models along with a stacking ensemble methodology to improve models' performance on extreme rainfall events five to 100 minutes in advance.

LSTM models, in particular, were popular among modelers for short-term rainfall prediction. [117] made use of the LSTM model to predict Cloud-Top

Brightness Temperature (CTBT) images. These images were then fed to a FFNN to predict precipitation. The proposed method was compared with other extrapolation and NWP models for rainfall forecasting. The LSTM model coupled with FFNN gave the best results for short-lead time prediction (1-6 h in advance).

[118] devised a ConvLSTM sequence to sequence model for rainfall prediction based on radar maps. The forecasting window spanned one to six hours. The proposed model was compared with Fully Connected-LSTM (FC-LSTM) model and a radar extrapolation technique called Rover. It was found that ConvLSTM significantly outperformed both models thanks to its ability to capture spatio-temporal variations.

Based on the LSTM model architecture, [119] also proposed a model for rainfall forecasting called PredNet. The model takes as input maps of rainfall amounts and outputs frames for the next ten time steps with five minutes resolution. The proposed model was compared with the GRU model for both classification and regression tasks and was found to outperform the compared model. It was also reported that the model performed better at the classification task than the regression task.

To summarize, rainfall modeling has been extensively explored. Predictive approaches can be overall categorized into process-based and data-driven models. More specifically, data-driven models can be further divided into radar echo extrapolation techniques, statistical techniques, and ML-based methods. Out of the ML-based approaches, DL approaches were extensively used and explored. Figure 9 illustrates the proposed classification of rainfall predictive models across the literature.

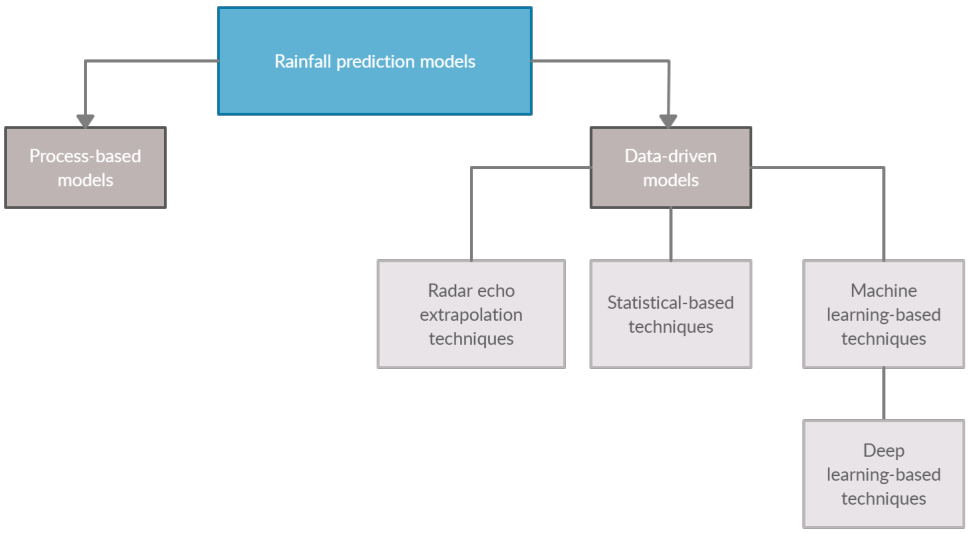

Figure 9: Rainfall prediction approaches classification.

Rainfall forecasting approaches can also be differentiated according to the input features used. While many used univariate models, i.e., only observed rainfall measurements were used as input, others resorted to multivariate modeling and used a plethora of meteorological and hydrological variables as predictors. Another essential factor that discriminates the presented approaches is the prediction lead time which ranges from minutes to years.

Limitations Several insights were derived from the presented literature review. First, there are several limitations to applying process-based models to short-term rainfall forecasting. These limitations include complex data assimilation techniques and requirements of a substantial amount of data with very high temporal and spatial resolution.

Statistical models were also extensively applied. In many studies, these models were unable to provide forecasts with high accuracy. ML and DL models were also extensively studied and compared for various rainfall prediction tasks. Many DL models showed promising results. Nonetheless, the suitability

of a model depends on the dataset under study. In many cases, simple models showed higher performance than sophisticated complex models.

Furthermore, it was found that nowcasting models were rarely investigated. Particularly models that are solely based on observed rainfall gauge measurements (due to data availability limitations). These models have significant practical importance as they can be integrated into hydrological models to improve forecasting [120].

2.2.3 Foundation work and Reseach Questions

This study stems from the project initiated by Erechtchoukova et al. which proposed a data-driven framework for flash flood prediction [22]. This framework uses readily available data recorded by water level and precipitation gauges to generate short-term flash flood predictions. This section summarizes the previously conducted work concerning this project.

In these studies, input data corresponded to various years with different hydrological characteristics (wet and dry). The proposed models were trained and tested separately for each year. A flash flood prediction model was developed based on the ensemble methodology. The ensemble consisted of five tree-based inducers: C4.5, CART, JRip, NNge, and REPTree. Multiple scenarios were investigated based on several combinations of input variables coming from different spatially distributed sensors. It was discovered that all rain gauges from the studied watershed provided important information. Moreover, it was stated that the ensemble provided reliable predictions up to 45 minutes lead time (the error was within the acceptable threshold imposed by operational flood risk management). On the other hand, individual inducers provided satisfactory performance only for up to 30 minutes.

One of the most significant issues encountered in data-driven flood modeling is the data imbalance problem. Naturally, the number of flood events is considerably smaller than non-flood events. Consequently, data-driven models have higher performance in predicting the majority class (non-flood tuples). To address this issue, [121] proposed three data manipulation strategies. The first consisted of oversampling high flow events tuple for each modeled year. The second strategy considered training a single model based on the wettest year and using it to predict flood events in other years. The last approach concerned injecting flood events from different years into each year's training set. It was found that the third strategy improved the performance in wet years and had a satisfactory performance in the dry ones.

To enhance the model performance, other aspects of the flash flood modeling problems were as well investigated. For example, in [122], Erechtchoukouva et al. studied the effect of aggregation rules on an ensemble model. The researched combination rules included: majority vote, maximum probability, minimum probability, average probability, and product of probabilities. It was stated that The minimum probability aggregation rule outperformed its counterparts. In another study, [123] highlighted the effect of data granularity on the predictive performance of the flash flood prediction model. Datasets with granularity varying from 15 to 60 minutes were used with different forecast horizons. It was pinpointed that the model delivered the best performance when the data granularity matched the prediction lead time.

This study aims to further enhance the predictive model performance for extended lead times by introducing precipitation forecasts and assessing their impact. Therefore our goal is to answer the following research questions:

1. **RQ 1:** Which data-driven model provides reliable and efficient rainfall

nowcasts?

2. **RQ 2:** What is the impact of introducing rainfall forecasts into the data-driven flash flood prediction framework?

Chapter 3

Methodology, software selection and dataset

3.1 Methodology: Knowledge Discovery in Databases (KDD) and the proposed framework

Knowledge Discovery in Databases (KDD) refers to the process of extracting knowledge from large datasets [8]. The KDD process is iterative and consists of several stages (Figure 10).

The first step of the KDD process is domain understanding and goals outlining. This step requires understanding the problem domains and their underlying concepts. This study concentrated on the development and investigation of modeling techniques for rainfall nowcasting and short-term flood forecasting. The second step corresponds to the data selection and collection process. Prior to the patterns discovery process, the data need to go through cleaning and preprocessing. The third stage includes data cleaning and transformation techniques such as missing values imputation, scaling (aggregation), *etc*. The subsequent

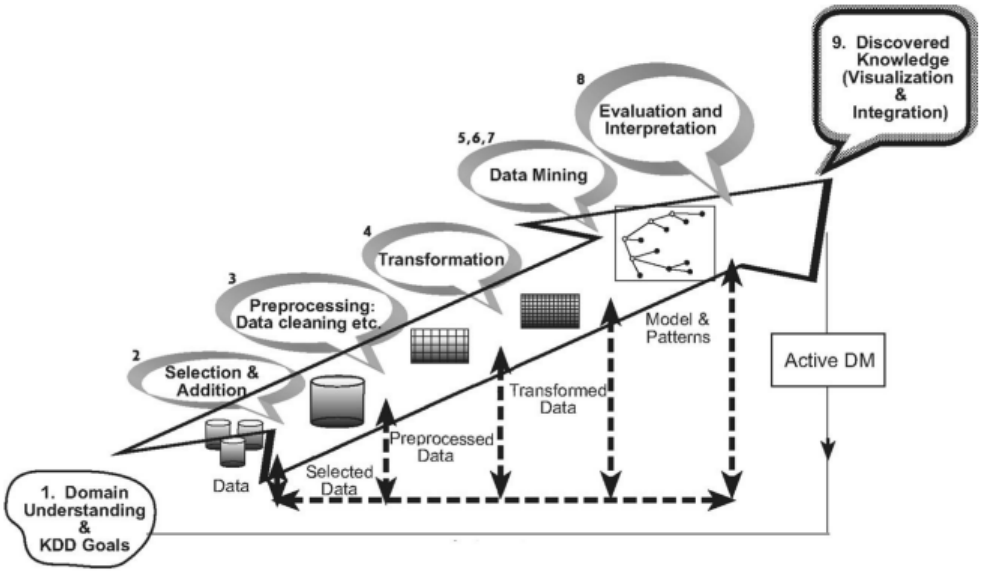

Figure 10: KDD process steps [8].

step corresponds to data mining. By definition, data mining is the process of prospecting large amounts of data to find patterns. This phase involves selecting the appropriate ML algorithm for the modeling task. For this project, data mining is used to predict rainfall values and classify flash flood events. Next, models evaluation and results interpretation is conducted. The last step concerns integrating the discovered knowledge into real-world applications.

Further, we propose a modeling framework that integrates rainfall nowcasting and assesses the impact of rainfall predictions uncertainties on a DL-based flash flood prediction model. In the proposed framework, we iterate through the KDD process multiple times. The framework consists mainly of four major steps (Figure 11). The first step concerns generating reliable rainfall nowcasts based on readily available rainfall sensor data. The second element of the framework quantifies the rainfall nowcasts uncertainties. We considered two scenarios for uncertainty quantification. First, we modeled uncertainty in a simple way by

integrating errors into future observed rainfall values. The second scenario consisted of quantifying the DL-based rainfall model uncertainty. In the third step, we propose a DL-based model that integrates rainfall forecasts along with past rainfall and water level values to predict flash flood occurrence. In the last step, the effect of rainfall uncertainties is studied.

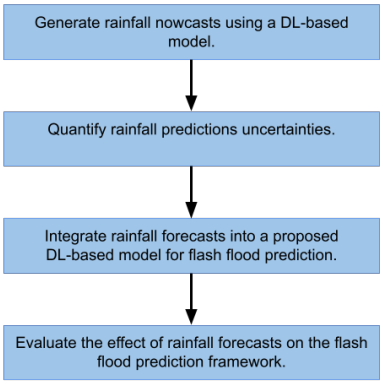

Figure 11: DL-based framework for flash flood prediction that support rainfall predictions integration and assessment of uncertainties.

3.2 Software selection

The data science field has gained immense popularity due to the rapid expansion of open-source software for knowledge discovery and ML, which are accompanied by numerous tutorials available online. Python, R, Matlab, SAS, SPSS, and STATA are the most popular [124, 125]. Each of these software tools has its srengths. Matlab, SPSS, SAS, and STATA offer various packages and libraries for data analysis and modeling. Python is a general-purpose programming lan-

guage that is open source. This software tool has vast community support. R is a statistical computing software that offers abundant libraries and packages for statistical analysis [125].

[126] studied several data science tools and reported that Python was the most easily learned, read, and used programming tool. In this study, Python V 3.7.12 along with Google Colab Integrated Development Environment were chosen for the rainfall and flash flood modeling problems.

In the context of our project, a set of statistical and DL models were applied. For the statistical modeling, we used Python 'statsmodels' package. This package offers a variety of time-series data analysis functions and models. The most popular DL tools that offer a python interface are Theano, Caffe, Deep Learning 4j (DL4J), Tensor Flow (TF), CNTK, PyTorch, and MXNET [127, 128]. Each of these tools offers a plethora of DL models ranging from simple deep neural networks to more sophisticated architectures such as CNNs, RNNs, and LSTMs. However, they portray different characteristics in terms of ease of use, flexibility, community support, scalability, portability, and application domains.

A summary of some of the principal characteristics of these frameworks is presented in Table 3.1.

Table 3.1: Deep learning frameworks comparison.

Framework	**Release year**	**Core language**	**Community support**	**Open Source**	**Use cases**
Theano	2007	Python	Development ceased	✓	RNNs
Caffe	2013	C++	Limited	✓	Computer Vision
DL4j	2014	Java	Moderate	✓	Various
TF	2015	C++	Large	✓	Various
CNTK	2016	C++	Development ceased	✓	RNNs, CNNs
MXNET	2016	C++	Limited	✓	Various
PyTorch	2016	C++, Python	Moderate	✓	Various

TF has the largest community support. In fact, TF was ranked first on Github in terms of the number of projects [127]. This feature is crucial as it allows for

active development of the framework, ease of access to information, good quality documentation, and tutorials. Furthermore, a comparative study of different DL frameworks conducted by [127] showed that TF had the highest overall score in terms of various aspects compared to other frameworks. These aspects included model design ability, interface property, deployment ability, performance, framework design, and prospects for development. Therefore, we chose TF 2.7 to conduct our experiments. To simplify the development of ANN of various architectures, Keras 2.7 API was employed.

3.3 Dataset

Collected data correspond to rainfall and water level measurements recorded at the Spring Creek watershed, Ontario, Canada. This watershed has a surface of roughly 50 km and a slope of 5% [121]. It takes part in the Etobicoke Creek watershed that drains into Lake Ontario. According to the recently published Toronto and Region Conservation Authority (TRCA) report card [9], The Etobicoke creek watershed is highly urbanized. It consists of 67% urban area, 19% rural area, and 14% natural cover (Figure 12). This watershed is characterized by a short response time which is equal to 30 minutes. Due to the changes in land surface (increase in impervious surface and constructions), flooding increased notably.

The data correspond to rainfall measurements recorded at Heart Lake (HL) and Mississauga Works Yard (M) locations. The precipitations time series data has five minutes granularity. The water level data were recorded by stream gauges installed in the Spring Creek North (SN) and Spring Creek South (SS) locations (Figure 13). These data have 15 minutes resolutions. SS corresponds

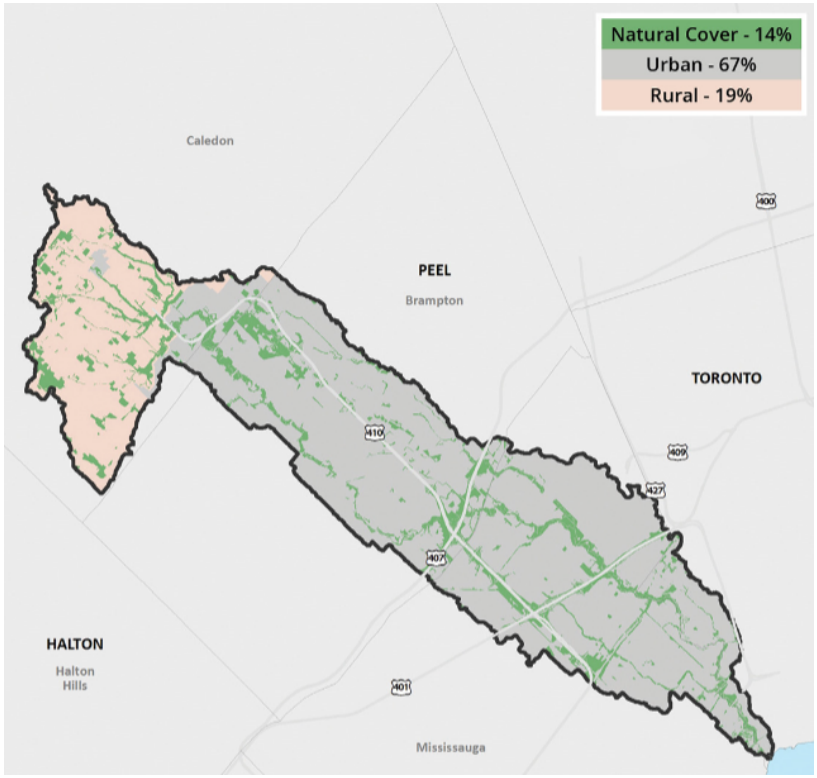

Figure 12: Etobicoke creek watershed land cover [9].

to the target location where we try to predict the occurrence of flash floods.

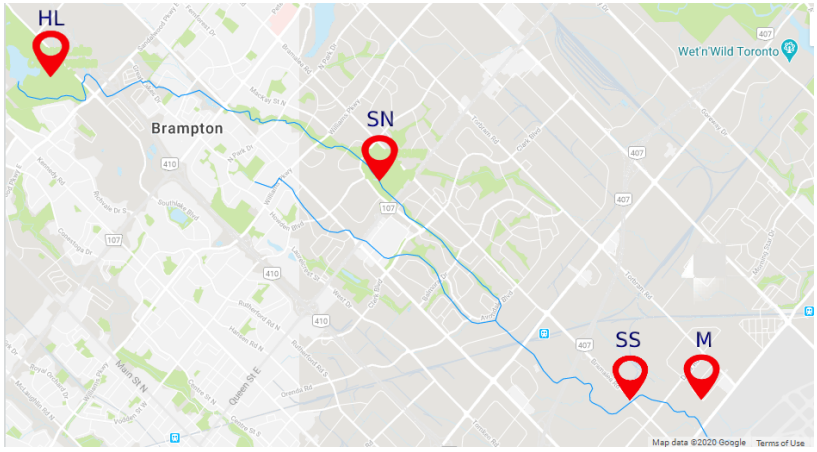

Figure 13: Sensors locations (TRCA gauging).

Datasets used in this study were collected over a warm period from April to December during the years 2013, 2014, 2015, and 2016. It is worth mentioning

that this period corresponds to snowmelt which causes increased water levels and, consequently, increased risk of flash flood occurrence. The studied years have distinct hydrological characteristics. The year 2013 was considered wet, while the remaining years were dry. These characteristics are well illustrated in Figure 14, which demonstrates the sum of rainfall amounts in mm per month. It can be clearly seen that 2013 had the highest monthly rainfall amounts compared to the others.

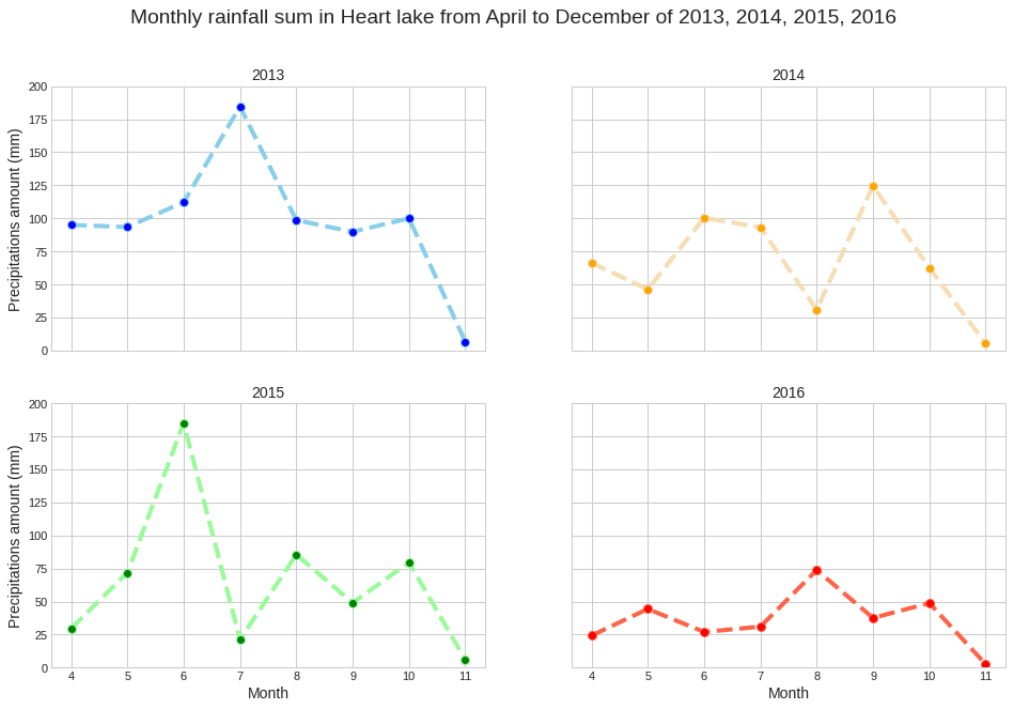

Figure 14: Monthly rainfall amounts in mm during the warm period of years 2013, 2014, 2015, and 2016.

Chapter 4

Data-driven rainfall nowcasting

Our objective is to find a data-driven rainfall nowcasting model with a satisfactory predictive ability based solely on data received from rain gauges. The modeling was conducted according to two scenarios. We investigated the performance of several data-driven regression models using single sensor data. Alternatively, we combined data gathered from neighboring sensors to predict rainfall at the target location.

4.1 The investigated models: Persistence, ARIMA, and LSTM

We investigated the performance of the following modeling approaches: a statistical model -ARIMA, a DL-based model -LSTM, and the persistence model.

Persistence model The persistence model, also referred to as the naïve model, is trivial. It assumes that atmospheric conditions will be the same at the next time step as they are observed now. Mathematically it can be expressed in the

following way:

$$Y_{t+leadtime} = Y_t \tag{4.1}$$

where Y_t is the rainfall amount at time t.

ARIMA model The ARIMA model is a popular statistical times series forecasting model. It consists of two components. The first component is a linear combination of observed values (deterministic). The second component is the sum of random errors (stochastic). Future values of a time series variable Y_t are predicted according to the formula presented in equation 4.2.

$$Y_t = \beta_1 Y_{t-1} + \beta_2 Y_{t-2} + .. + \beta_p Y_{t-p} + \varepsilon_t + \phi_1 \varepsilon_{t-1} + \phi_2 \varepsilon_{t-2} + .. + \phi_q \varepsilon_{t-q} \tag{4.2}$$

where ε_t is the random error at time t. $\beta_i \, (i = 1..p)$ and $\phi_i \, (i = 1..q)$ represent the model parameters. p is the autoregressive component order, and q is the moving average component order. It is important to note that depending on the order of differencing d; the variable Y_t would be differenced d times before being modeled using equation 4.2.

LSTM model the LSTM model has a RNN model architecture. The latter is a type of ANN architecture constructed to model sequential data. Unlike traditional ANN, RNN accounts for the dependencies between time-series data points by adding links between adjacent nodes within one single layer. In many studies, RNN models were found to deliver good performance for time series analysis [129]. Nonetheless, when the number of time steps increases (the case of long-term dependencies), this architecture is prone to the vanishing gradient

issue. That is, changes in later layers cannot be reflected in the preceding ones (the gradient vanishes as it goes back through time). Thus, RNN models are incapable of capturing long-term dependencies. The LSTM model was proposed to address this issue [130, 131]. The LSTM architecture is mainly characterized by its memory cells. This cell helps to conserve the flow of information from one unit to another. This characteristic gives the model the ability to capture and represent long-term dependencies. LSTMs are, therefore, capable of modeling both short-term and long-term dependencies, which makes them an attractive tool for time series modeling [132]. Figure 23 presents the LSTM cell architecture.

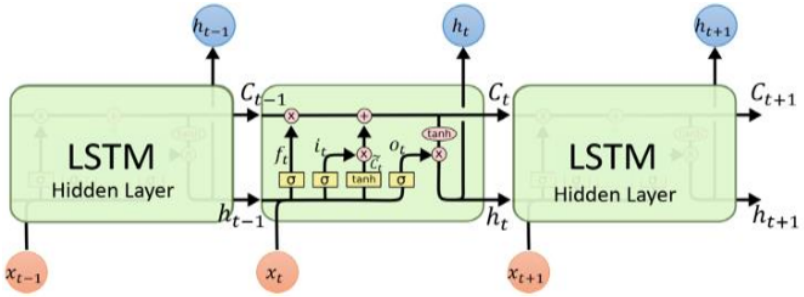

Figure 15: The LSTM model architecture [10].

where x_t represents the input, h_t represents the hidden state, c_t represents the final cell state, and $\tilde{c}_t$ corresponds to the new candidate state (based on which the cell state would be updated). f_t, i_t, and o_t correspond to the forget, input, and output gates, respectively. The forget and input gates control how much information from the past state and current input should be retained in the current state. The output gate controls how much of the current state should be retained for the output. Mathematical expressions corresponding to the presented architecture are displayed in the set of equations 4.3.

$$
\begin{aligned}
f_t &= \sigma(W_{fh}h_{t-1} + W_{fx}x_t + b_f) \\
i_t &= \sigma(W_{ih}h_{t-1} + W_{ix}x_t + b_i) \\
\tilde{c}_t &= \sigma(W_{\tilde{c}h}h_{t-1} + W_{\tilde{c}x}x_t + b_{\tilde{c}}) \\
c_t &= f_t.c_{t-1} + i_t.\tilde{c}_t \\
o_t &= \sigma(W_{oh}h_{t-1} + W_{ox}x_t + b_o) \\
h_t &= o_t.tanh(c_t)
\end{aligned}
\tag{4.3}
$$

where $W_{fh}, W_{fx}, W_{ih}, W_{ix}, W_{\tilde{c}h}, W_{\tilde{c}x}, W_{oh}, W_{ox}$ represent the weights of the units and $b_f, b_i, b_{\tilde{c}}, b_o$ represent the biases. It is important to add that in these formulae, the dot (.) corresponds to the element-wise multiplication.

4.2 Evaluation metrics

Several metrics (over 40) can be found in the literature for the evaluation of regression models [133]. There is no ideal metric that best describes a model performance. Each performance measure captures specific characteristics of the error. Therefore, the choice of appropriate evaluation metrics depends on the modeling problem.

In regression modeling, five evaluation metrics are traditionally used. These metrics are RMSE, Mean Absolute Error (MAE), Mean Absolute Percentage Error (MAPE), Symmetric Mean Absolute Percentage Error (SMAPE), R^2 (Nash–Sutcliffe), and Maximum Residual Error (MRE). Their formulae are presented in the set of equations 4.4, where Y_i corresponds to the observed rainfall amount, $\hat{Y}_i$ corresponds to the predicted rainfall amount, and $\bar{Y}$ presents the mean of the

observed values.

Each of these metrics has its own characteristics. RMSE penalizes large errors, MAE weights errors similarly (for example, an error of 2 is twice as much as an error of 1), MAPE is not defined for $y = 0$ and is not suitable for data with abrupt changes and peak values (error values would explode), SMAPE is insensitive to fluctuations in data, R^2 compares model's performance to a naive model (mean) and doesn't quantify the real error and, MRE doesn't summarize errors, it only takes the largest value of residuals.

According to the selected modeling goal, RMSE and MRE can be used to evaluate our models. RMSE penalizes significant errors while MRE gives insight into the largest error committed.

$$
\begin{aligned}
RMSE &= \sqrt{\frac{1}{n}\Sigma_{i=1}^{n}\left(Y_i - \hat{Y}_i\right)^2} \\
MAE &= \frac{1}{n}\Sigma_{i=1}^{n}\left|Y_i - \hat{Y}_i\right| \\
MAPE &= \left(\frac{1}{n}\Sigma_{i=1}^{n}\frac{\left|Y_i - \hat{Y}_i\right|}{|Y_i|}\right) * 100 \\
SMAPE &= \left(\frac{1}{n}\Sigma_{i=1}^{n}\frac{\left|Y_i - \hat{Y}_i\right|}{\frac{(|Y_i| - |\hat{Y}_i|)}{2}}\right) * 100 \\
R^2 &= 1 - \frac{\Sigma_{i=1}^{n}\left(Y_i - \hat{Y}_i\right)^2}{\Sigma_{i=1}^{n}\left(Y_i - \bar{Y}\right)^2} \\
MRE &= MAX_i(|Y_i - \hat{Y}_i|)
\end{aligned}
\tag{4.4}
$$

4.3 Experimentation

4.3.1 Data preprocessing

The first data preprocessing step consisted of identifying the missing and erroneous values in the precipitation time series. Further analysis revealed that

these values corresponded to dry periods and were due to gauge malfunctions. Therefore, the faulty records were substituted with zero values. The used time series were characterized by a fine granularity (five minutes). In this study, up-scaling was performed to transform the temporal resolution from five minutes to 15 minutes. It is worth mentioning that the choice of the scale was made based on the hydrological characteristics of the Spring Creek and its watershed.

4.3.2 Experiments description

Overall, the established set of experiments can be divided into univariate-modeling approach experiments and multivariate-modeling approach experiments. In both groups of experiments, separate models for lead times ranging from 15 minutes to 60 minutes were established. Further, data corresponding to each year were chronologically split into training and testing sets with a 70%:30% split ratio.

Experiments set 1: univariate modeling

This set of experiments consists of predicting rainfall at each target location using data corresponding to that location solely.

The persistence model was directly applied to the test set, as it does not involve any model training.

To create the ARIMA estimator, firstly, the data were checked for stationarity. ACF and PACF plots were produced and examined to get insights into the maximum values of the moving average and the autoregressive parameters, q and p, respectively. The models were then created for different combinations of q and p while taking into account their maximum values. The best-performing model was selected according to the Akaike information criterion, which is a statistical

measure of the goodness of fit of an estimator. Figures 16 and 17 represent the ACF and PACF plots of rainfall time series at the HL location observed during 2013. Based on the number of significant spikes in the ACF and PACF plots, a maximum value of p was set to 2, and a maximum value of q was set to 10.

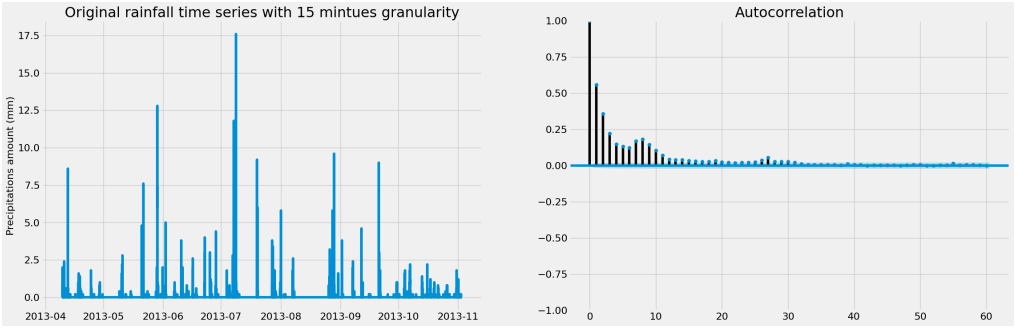

Figure 16: Original rainfall time series and ACF plots corresponding to HL sensor, year 2013.

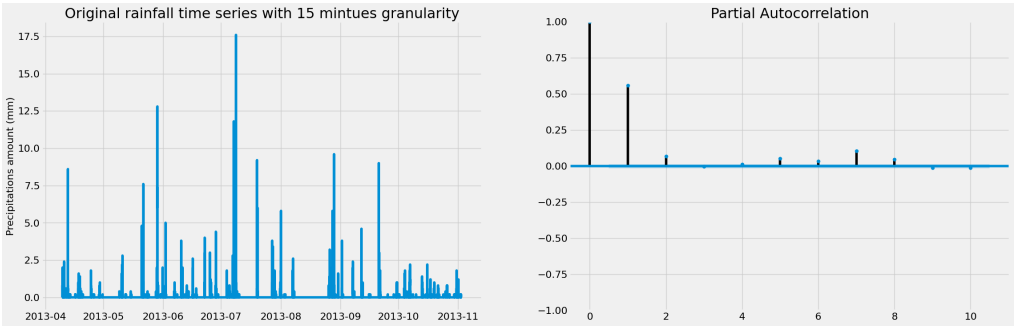

Figure 17: Original rainfall time series and PACF plots corresponding to HL sensor, year 2013.

The selected model performance was assessed on the holdout test sample. The Python package 'statsmodels' was used to automate this procedure.

To develop the LSTM models, several steps were performed. The time series values were first normalized using the MinMax scaler. Obtained data were then transformed into a format of input-output variables, wherein the input variables represented lagged values of rainfall prior and up to time t, and the output variable represented precipitation magnitude at the prediction time (t + number of

prediction time steps). Each training tuple had the scheme presented in formula 4.5.

$$(x_{t-n\tau}, .., x_{t-2\tau}, x_{t-\tau}, x_t \; ; \; x_{t+m\tau}) \tag{4.5}$$

where n represents the number of lag steps, m represents the number of lead time steps, and τ corresponds to the length of each time step (15 minutes in our experiments).

To determine the suitable number of the lagged values, insights from ACF and PACF plots were taken into account. It is essential to mention that the input data were transformed into a multidimensional array with the following format: [number of records, number of time steps, number of features] as required in all of the LSTM models' implementations. In our experiments, the shape of the input array = [number of records, 12, 1]. i.e., the number of time steps equals 12 (12*15 minutes = 180 minutes = three hours), and we have one feature (rainfall measurements at HL or M location).

To train the LSTM model, we followed the empirical procedure recommended in [134]. First, we started by training a simple version of the model. Then, we gradually increased the model complexity in terms of the number of units and layers. Once the model overfitted the data, we applied the recurrent layer dropout regularization technique. Once we found a suitable model, we performed hyperparameter tuning using hyperparameter values within the range of the model's hyperparameters established beforehand. The hyperparameter search was done using a Bayesian optimization search algorithm.

The architecture consisted of several stacked LSTM hidden layers and a fully connected dense layer. Various hyperparameters were considered. We ranged the number of layers and the number of units in each layer. We also

considered several values for the learning rate, batch size, and the number of epochs. RELU, SIGMOID, and TANH were investigated as possible transfer functions. Furthermore, the Gradient Descent (GD)-based ADAM optimizer was used for model training.

It is worth mentioning that depending on the year, different combinations of hyperparameters were found suitable. All the models consisted of one LSTM layer with a number of units varying between two and five. The dense layer consisted of one neuron. Further, we have used a learning rate of 0.01. The batch size varied between 15, 32, and 64. The models were trained for 200 and 300 epochs depending on the year. Finally, we evaluated the best model performance on the hold-out testing set.

Experiments set 2: multivariate modeling

The LSTM model was chosen to conduct all the multivariate modeling approach experiments. Multi-sensor data were fed into the model, i.e., data from both HL and M were used to predict rainfall at HL and M locations. Each training tuple can be represented according to the scheme illustrated in equation 4.6.

$$(y_{t-n_1\tau},..,y_{t-\tau},y_t,x_{t-n_2\tau},..,x_{t-\tau},x_t;x_{t+m\tau}) \tag{4.6}$$

where x_t and y_t represent precipitation magnitudes at different locations, n_1 and n_2 represent the number of lag steps corresponding to each sensor. m and τ can be defined as previously.

ACF and PACF plots were used to determine the appropriate number of lags for the target location data. The number of time delay steps used for the adjacent sensor was chosen depending on the distribution of the time difference between the start of rainfall events at the neighboring and target locations. Figure 18

depicts this distribution considering that HL represents the target location and M represents the adjacent location. It can be seen that the majority of steps are inferior to ten. Therefore, ten lag values were used from the adjacent sensor (in this case, M).

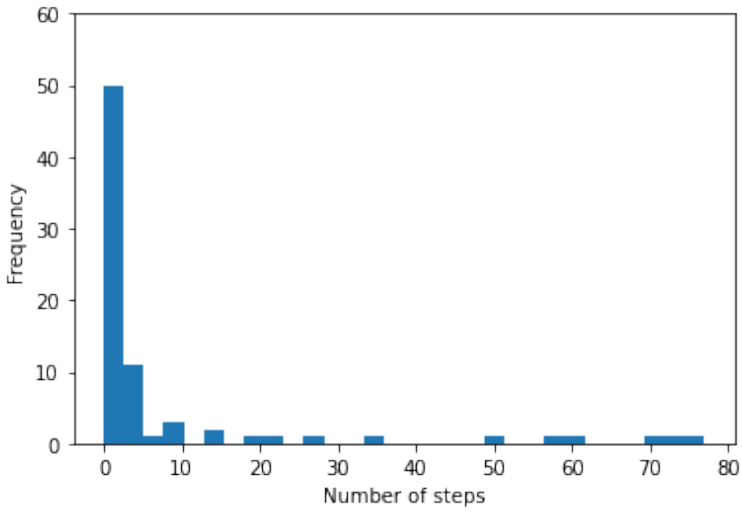

Figure 18: The distribution of rainfall start time difference between HL and M using 2013 data.

4.4 Results and discussion

Figure 19 presents the predictive ability of the investigated models in terms of RMSE on HL location using HL data solely. Overall, the persistence model delivered the worst performance. The two other models gave inconsistent results for different modeled years. While in 2013, the LSTM model significantly outperformed the other models with a decrease in RMSE reaching roughly 30%, in the other years, results were different. In 2014 and 2016, the LSTM model delivered comparable results to the ARIMA model. However, the LSTM model had the most significant error on data corresponding to the year 2015. Further analysis of the hydrological characteristics of the precipitation in 2015 showed salient

discrepancies in rainfall distributions between the training and the testing sets periods. The model was thus unable to learn all the hydrological patterns due to the non-representativeness of the training set. Overall, even though the LSTM model yielded superior performance in some years, its superiority cannot be generalized and largely depends on the patterns in the hydrological conditions within a single year. More focus should thus be shed on data representativeness rather than model sophistication.

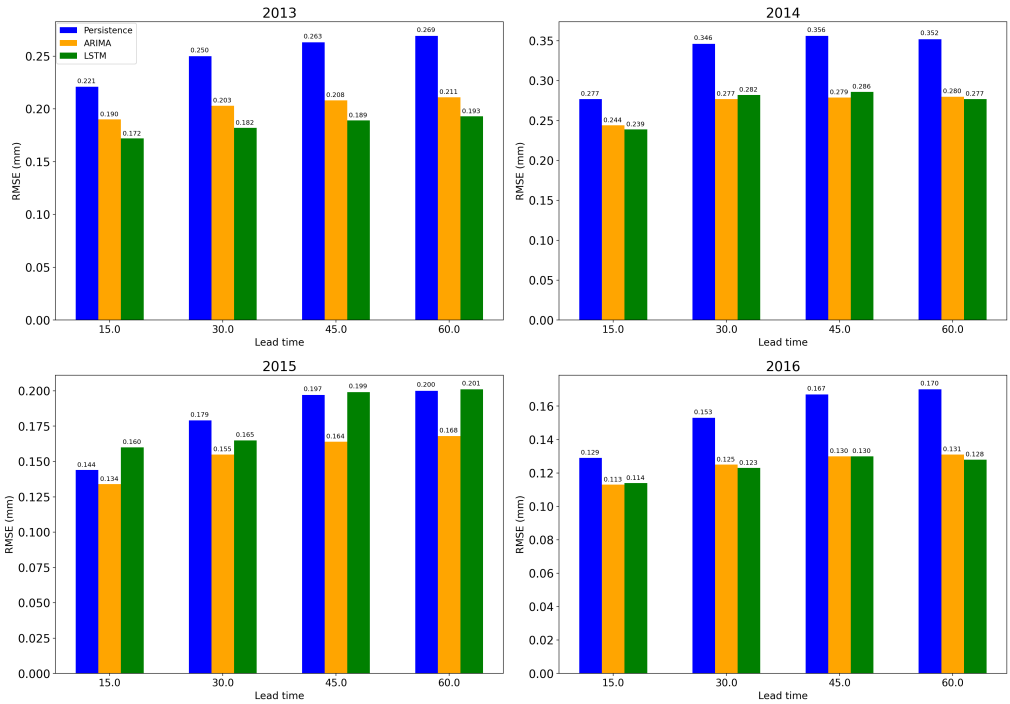

Figure 19: RMSE of Persistence, ARIMA, and LSTM models of the rainfall prediction at HL location using data from the same sensor.

As opposed to the RMSE, when the MRE evaluation metric was considered, it was found that all the models performed comparably the same as the naïve model (around 9 mm for 2013, 12 mm for 2014, 4 mm for 2015 and 2016), as shown in Figure 20. Error analysis showed that the models erroneously predicted extreme peak values which occurred at the beginning or end of the rainfall events.

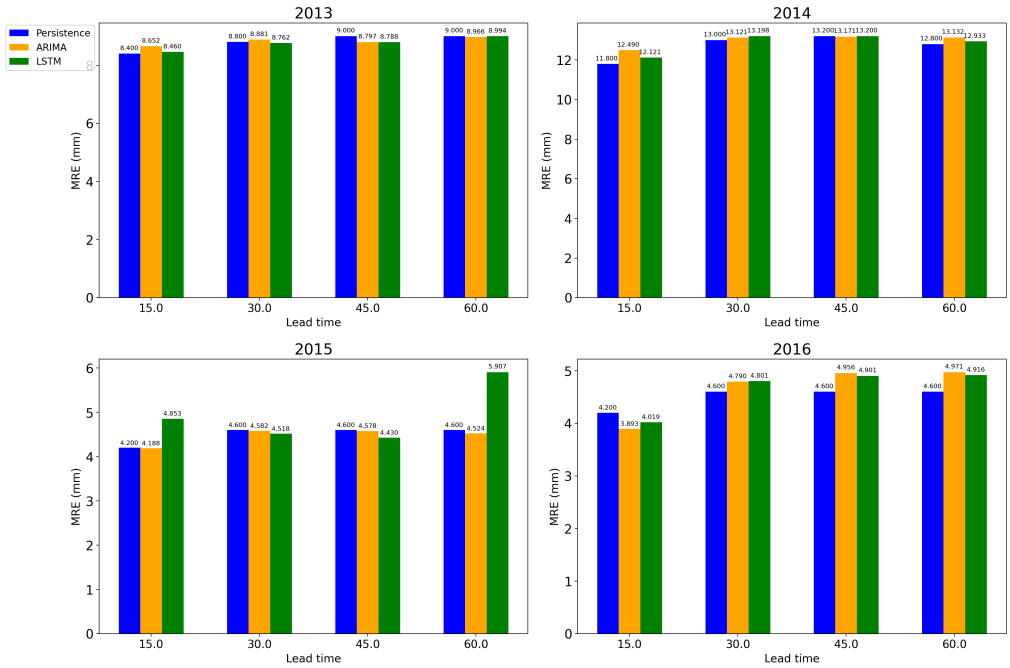

Figure 20: MRE of Persistence, ARIMA, and LSTM models of the rainfall prediction at HL location using data from the same sensor.

Figures 21 and 22 present the MRE obtained from the multi-sensors approach models. Figure 21 compares the rainfall prediction error at the HL location while taking as input data from HL rain gauge only or data from both M and HL ground stations. Figure 22 presents the results for rainfall prediction at the M location. As opposed to the results obtained from the first set of experiments (uni-variate approach), and depending on the prediction year and location, the MRE of the LSTM model showed major improvement. Overall, combining measurements from both stations enhanced the models' predictive ability in both locations for 2014 and 2015. Nevertheless, while in 2013, integrating data from several gauges significantly increased the performance for M location, for HL location, the performance dropped. Combining 2016 data from both gauges failed to enhance the performance as well. This fact suggested looking into dominating meteorological conditions of the watershed in differ-

ent years. Archived data obtained from Environment Canada's official website showed that records for 2013 mostly represent winds blowing from north to south. Given that the HL is located in the northern part of the investigated watershed and the M gauge is installed on the southern part, the improved prediction at the M location while data from both gauges are utilized can be easily justified.

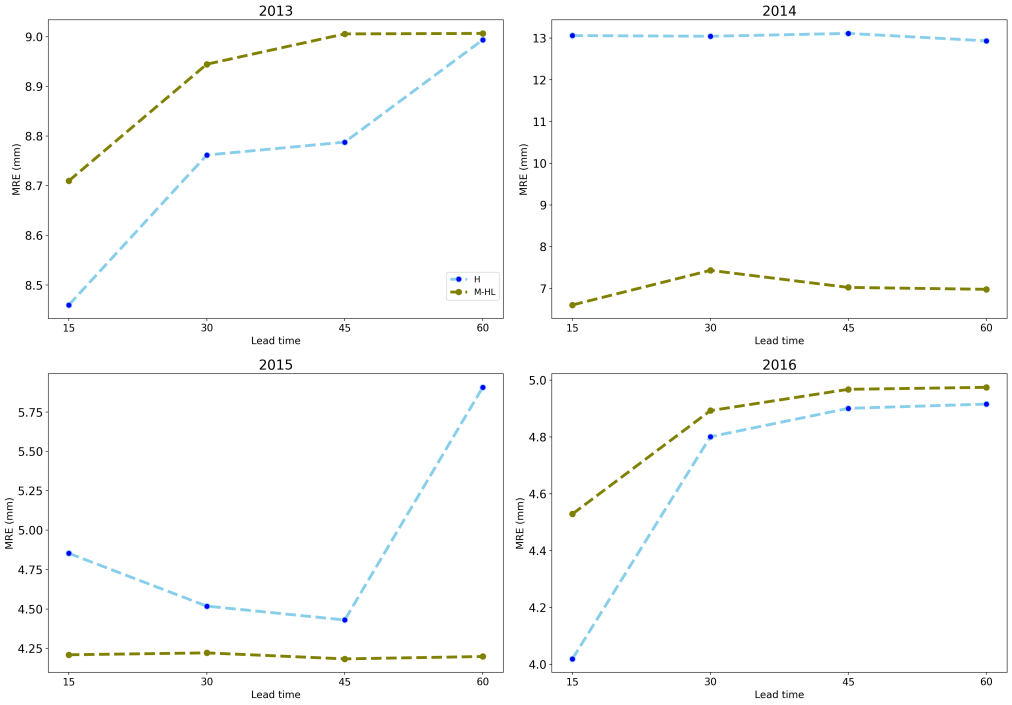

Figure 21: LSTM rainfall prediction model performance at HL location using data solely from HL sensor and a combination of both datasets from HL and M.

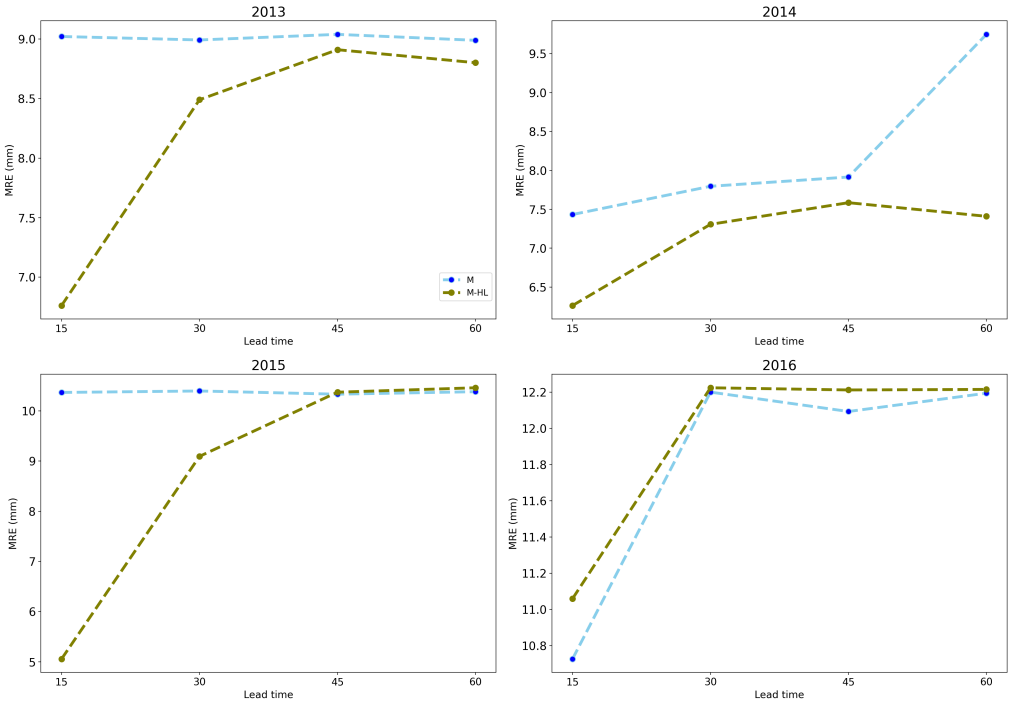

Figure 22: LSTM rainfall prediction model performance at M location using data solely from M sensor and a combination of both datasets from HL and M.

Obtained results reveal that combining data from multiple sensors along with the LSTM model gave promising results. Thus, more focus should be placed on data representativeness and richness. In the next step, we will focus on building the flash flood prediction model, estimating the uncertainty of predicted rainfall magnitudes, and studying the effect of integrating rainfall forecasts into the flash flood prediction model.

Chapter 5

Data-driven flash flood modeling and the effect of rainfall nowcasting uncertainty

5.1 Models design

The LSTM models were extensively and successfully used for flood prediction. Particularly, they were applied for regression tasks such as streamflow and water level magnitudes forecasting. We chose to investigate their performance on the flash flood classification problem.

Overall, the experiment sets conducted in the context of this project can be divided into two main classes. In the first class, only past observed values of the input variables were used as input. The second class of experiment sets involved integrating predicted rainfall data into the framework to forecast hydrological events in the near future. For the first class of experiments, a traditional LSTM model consisting of one LSTM layer, one densely connected layer, and

one output layer consisting of one single node were chosen for the flash flood prediction problem.

The LSTM model requires the input data to have a specific shape with the same number of time steps for all the input features. This condition is met in the first class of experiments. However, this is not the case for the second class of experiments, where the number of rainfall variables vary depending on the lead time. Moreover, further research showed that no existing LSTM model architecture supports input features with varying numbers of time steps. Following [69], we apply an Encoder-Decoder LSTM-based model for the second class of experiments.

Figure 23 depicts the architecture of the LSTM-based model used for the flash flood prediction task using only the most recent observations. To determine the model hyper-parameters set, we followed a similar procedure as in the data-driven rainfall modeling task. Firstly, we found a set of hyper-parameters with good predictive performance through trial and error and random search. Secondly, we applied Bayesian hyperparameter optimization on a set of hyper-parameters' values within the range of the previously determined values.

Depending on the modeled year, the LSTM layer consisted of 16 or 18 units. The dense layer comprised 16 neurons with the RELU activation function, and a Sigmoid activation was used for the output neuron. The model was trained for 25 or 30 epochs using the Binary Cross-Entropy loss function and a learning rate of 0.01.

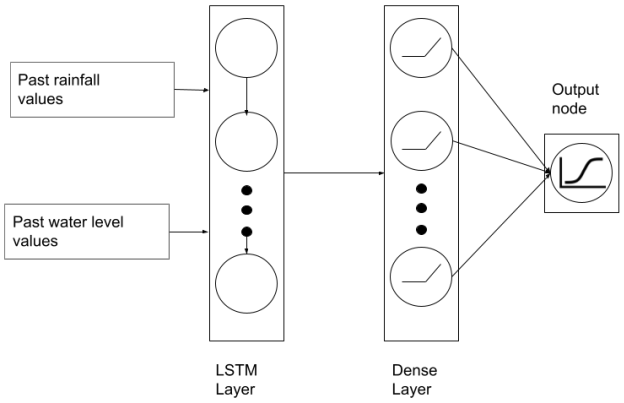

Figure 23: The simple LSTM model architecture.

The introduction of rainfall predictions results in a different number of elements corresponding to the precipitation variables and water level variables. The Encoder-Decoder LSTM-based model was inspired by the GRU-based sequence to sequence model, denominated Neural Runoff Model (NRM), established by [69] for runoff predictions. We alter the proposed architecture and suggest a model suitable for events classification rather than sequence to sequence modeling. Figure 24 depicts the architecture of the proposed model. The 'encoder' part of the model encodes the past input data into a latent space. The encoding is presented by the hidden state and a cell state vectors. These states are then fed as the initial state to the decoder, which takes the predicted rainfall magnitudes as input. We can say that the latent representation plays the role of a context to the decoder model. For comparison reasons, the encoder and decoder LSTM layers had the same configuration as the LSTM-based model previously described. The densely connected layer and the output layer attached to the decoder had as well the same configurations as the LSTM-based

model. Further, we have used the same set of hyperparameters.

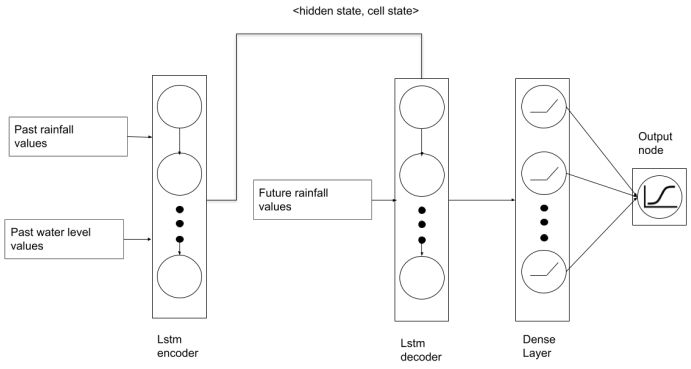

Figure 24: The Encoder-Decoder LSTM-based model architecture.

5.2 Evaluation metrics

Several metrics were proposed to appraise the performance of data-driven classification models. Depending on the problem domain and the classification task, different performance indicators might be found suitable.

The flash flood prediction data is naturally characterized by a severe data imbalance as the number of flood events is significantly lower than the number of non-flood events. Due to this imbalance, almost all data-driven models demonstrate near to perfect performance in classifying non-flood events and introduce the majority of errors in predictions of flood events. In addition, from a practical perspective, correctly identifying flash flood events is more critical than non-flood events. Therefore, we will focus on the models' ability to detect flash flood events (True positives).

Three binary classification assessment metrics are used: recall (also referred to as sensitivity), precision, and F1 score.

$$Recall = \frac{TP}{TP + FN} \tag{5.1}$$

$$Precision = \frac{TP}{TP + FP} \tag{5.2}$$

$$F1_{score} = \frac{2 * Precision * Recall}{Precision + Recall} \tag{5.3}$$

where:

- TP (True Positives): The number of flood events that were correctly identified.
- FP (False Positives): The number of non-flood events that were incorrectly predicted as flood (false alarm error).
- TN (True Negatives): The number of non-flood events that were correctly classified as non-flood.
- FN (False Negatives): The number of flood events that were incorrectly classified as non-flood (non-detection error).

5.3 Experimentation

5.3.1 Data prepossessing

Similar to the rainfall time series, the water level data underwent a set of data cleaning and preprocessing steps. First, we identified the missing and erroneous records which were filled with the mean value. Next, all the processed precipitation and water level time series were brought to the same granularity (15 min-

utes), synchronized, and transformed into tuples consisting of input variables and the target variable. The input variables correspond to lagged magnitudes of rainfall and water stages recorded in different locations. The target variable specifies the occurrence of flood in the cross-section of interest at a certain lead time, with the value 0 indicating non-flood and the value 1 indicating the occurrence of flood. Each Tuple had a format illustrated by the formula 5.4.

$$(x_{l1w,t-n\tau},...,x_{l1w,t},x_{l2w,t-n\tau},...,x_{l2w,t},y_{l1p,t-n\tau},...,y_{l1p,t},y_{l2p,t-n\tau},...,y_{l2p,t};o_{t+m\tau}) \tag{5.4}$$

where $x_{lw,t}$ corresponds to the water level values at locations l at time t (including the target location l_{target}). $y_{lp,t}$ represents precipitation values recorded by rainfall sensors at the moment t. n is the number of lag steps, and τ is the duration of each time step. $o_{t+m\tau}$ specifies whether the tuple corresponds to a flood event or not. In this case study, $\tau = 15\,min$ and the number of steps $n = 12$. These numbers were selected based on an analysis conducted by [22], where it was found that a three-hour delay ($15\,minutes * 12$) provided satisfactory model performance. Values of $o_{t+m\tau}$ are determined based on the following function:

$$o_{t+m\tau} = \begin{cases} 1, & \text{if } x_{l_{target},t+m\tau} >= threshold \\ 0, & \text{otherwise} \end{cases} \tag{5.5}$$

where $x_{l_{target},t+m\tau}$ is the water level at the cross-section of interest at the specified lead time $t+m\tau$. The TRCA Flood Management Team set the threshold value at 172.75 m for the investigated location.

5.3.2 Experiment sets'

The conducted experiments can be divided into four sets. The first set of experiments corresponded to applying the simple LSTM-based model using historical values of the input features. The second set consisted of integrating predicted rainfall magnitudes where future observed values were used as perfect predictions. In this set of experiments, we applied the suggested encoder-decoder model. In the third set of experiments, we introduced magnitudes and time errors into the future observed rainfall values to emulate rainfall forecasts errors and study their impact on the model's performance. The last set consisted of integrating the data-driven precipitation predictions generated by the previously established rainfall-forecasting model taking into account their uncertainty.

It is worth mentioning that to mitigate the data imbalance issue, we assigned different class weights to the flood and non-flood classes, with bigger weights given to the flood class records. This technique was found effective as it allows the model to pay more attention to the minority class by largely penalizing its errors. It is important to mention that for each year and lead time (ranging from 15 minutes to two hours), a separate model was established, as years have different hydrological characteristics. Each dataset corresponding to a year was chronologically split into training and testing with a 70%:30% split ratio. A specific function was created to transform the input data into a format acceptable by the LSTM-based models. The input data format is a three-dimensional tensor with the first dimension representing the number of records, the second dimension corresponding to the number of time steps of the input features, and the third dimension being the number of features (Figure 25). The input data were normalized using the Min-Max scaler because this technique is essential to speed up the convergence of the learning algorithm [135].

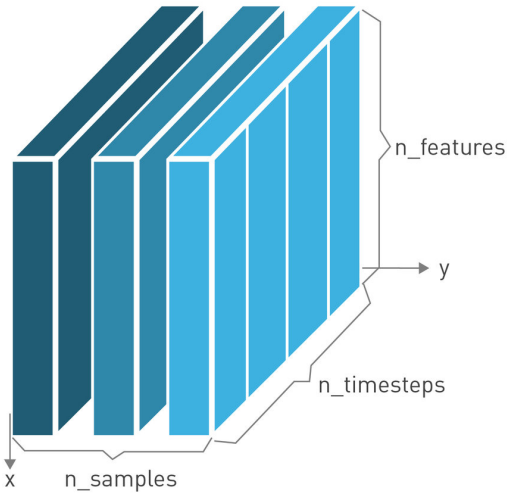

Figure 25: The LSTM model input shape [11]

Due to various sources of randomization (weight initialization, atomic addition operation, which vary depending on the order of addition and the rounding up of floating-point operations), reproducibility of DL-based models becomes very challenging. Particularly, the Keras framework with the TensorFlow backend suffers from uncontrollable randomization that cannot be addressed by setting the seed values [136]. Consequently, to compare the models' performance statistically, we conducted 30 repeated training and testing runs for each model.

The first set of experiments corresponds to the first class of experiments. It consisted of applying the simple LSTM model to past observed feature values to classify each tuple. The remaining experiments correspond to the second class. The second batch of experiments consisted of integrating the future observed rainfall values using the encoder-decoder model. In the third set of experiments, to simulate rainfall forecasts, we injected errors into the magnitude of future observed precipitation values and the time of rainfall events occurrence. Magnitude errors were introduced by adding and subtracting 20% of the future precipitation values. The time errors were represented by shifting the future pre-

cipitation magnitudes backward (-15min) and forward (+15min). We examined the impact of each error separately and in combination with the other errors. Table 5.1 presents a summary of the experiments we implemented in this set.

Table 5.1: The third set of experiments.

Experiment n	**+20% magnitude error**	**-20% magnitude error**	**shift forward time error**	**shift backward time error**
1	✓			
2		✓		
3			✓	
4				✓
5	✓		✓	
6	✓			✓
7		✓	✓	
8		✓		✓

In the fourth experiment set, we assessed the effect of the data-driven rainfall forecasts uncertainty on the performance of the Encoder-Decoder LSTM model. First of all, rainfall forecasts and their uncertainty estimates were generated. To do so, we applied the LSTM model presented in chapter four to produce the rainfall nowcasts. A few modifications were performed to the model's architecture to account for the prediction uncertainty. To evaluate the heteroscedastic aleatoric uncertainty, the following approach was adopted. The output corresponding to certain input x_i is considered as a Gaussian distribution with mean $\hat{\mu}_i$ and a variance $\hat{\sigma}_i{}^2$ instead of a scalar estimate y_i. The variance corresponds to the inherent randomness in the data. An extra node is then added to the output layer to estimate the variance magnitude. To account for this modification, instead of optimizing the mean squared errors loss, we minimize the Negative Log Likelihood loss function represented in Equation 5.6.

$$-\log p_\theta(y) = \frac{\log \hat{\sigma}^2}{2} + \frac{(\hat{\mu} - y)^2}{2\hat{\sigma}^2} \tag{5.6}$$

Where θ corresponds to the model parameters. Further to model the epistemic uncertainty, the MC dropout technique is activated during testing and multiple forward passes T are done for each input record x_i to obtain T outputs with the format presented in Formula 5.7.

$$\hat{\mu}_t, \hat{\sigma}_t{}^2 = f_\theta(x; w_t) \tag{5.7}$$

Where f_θ corresponds to the Neural Network model with parameters θ and w_t is the dropout mask at time t.

The final output $\hat{y}_f$ and the total variance $\hat{\sigma}_f^2$ are then calculated following Equations 5.8 and 5.9.

$$\hat{y}_f = \frac{1}{T}\sum_{i\in T}\hat{\mu}_t \tag{5.8}$$

$$\hat{\sigma}_f^2 = \frac{1}{T}\sum_{i\in T}\hat{\mu}_t{}^2 - \left(\frac{1}{T}\sum_{i\in T}\hat{\mu}_t\right)^2 + \frac{1}{T}\sum_{i\in T}\hat{\sigma}_t{}^2 \tag{5.9}$$

The calculated rainfall predictions were synchronized and introduced into the flash flood prediction model. To examine the impact of the rainfall prediction uncertainty, three scenarios were considered. In the first scenario, the rainfall point estimates predicted by the rainfall model (which is the mean of the T forward passes $\hat{y}_f$) were introduced (For example, for the 75 minutes flash flood forecast horizon, the mean predictions obtained from the 15, 30, 45, 60, and 75 minutes rainfall nowcasting models were used). In the second and third scenarios, the mean+2 Standard Deviation ($\hat{y}_f + 2\hat{\sigma}_f$) and the mean-2 Standard Deviation ($\hat{y}_f - 2\hat{\sigma}_f$) were introduced, respectively. It is important to mention that T was set to 100 forward passes. Moreover, we established separate models for lead times ranging from 15 minutes to two hours with 15 minutes time step.

Further, these experiments were conducted for the year 2013 solely, and only the univariate rainfall modeling technique was used due to computational time constraints.

5.4 Results and discussion

The first and second experimental sets showed that introducing accurate rainfall future values along with the Encoder-Decoder model significantly enhanced flash floods detection. Figures 26, 27, and 28 depict Box plots that summarize the recall values (corresponding to 30 runs) obtained before and after introducing precipitations for the years 2013, 2014, and 2015. It is worth mentioning that the box limits in the box plot correspond to the first quartile (Q1), median, and third quartile (Q3). While the lower and upper whiskers correspond to $(Q1 - 1.5 * (Q3 - Q1))$ and $(Q3 + 1.5 * (Q3 - Q1))$ respectively. The values which are outside this range are considered outliers.

Experiments in the year 2016 were not considered as the number of flood events in the testing set was negligible.

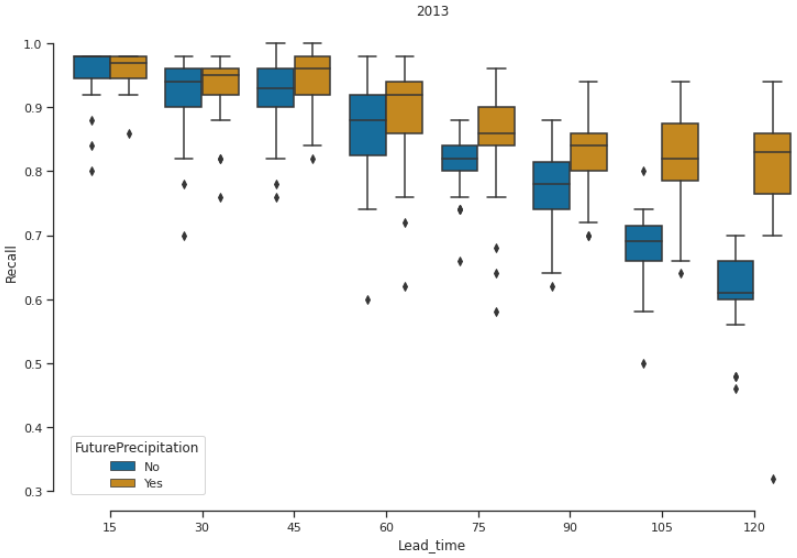

Figure 26: Box plots depicting recall values corresponding to the year 2013 (30 runs).

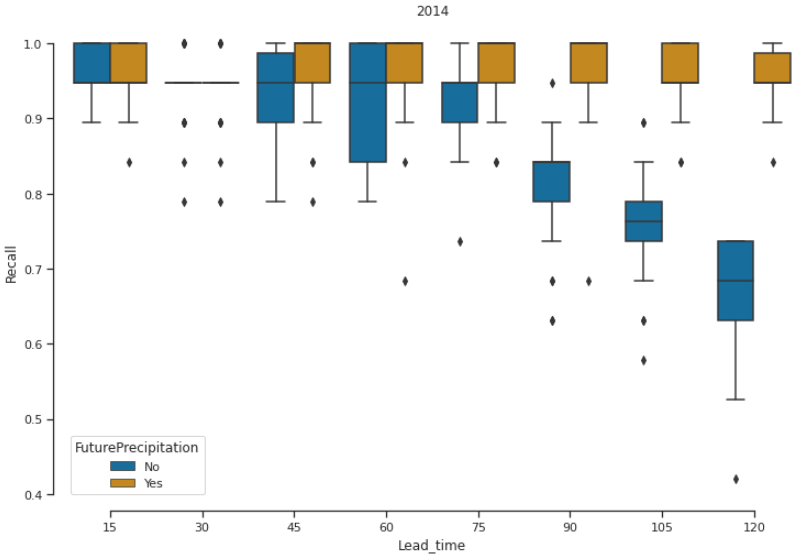

Figure 27: Box plots depicting recall values corresponding to the year 2014 (30 runs).

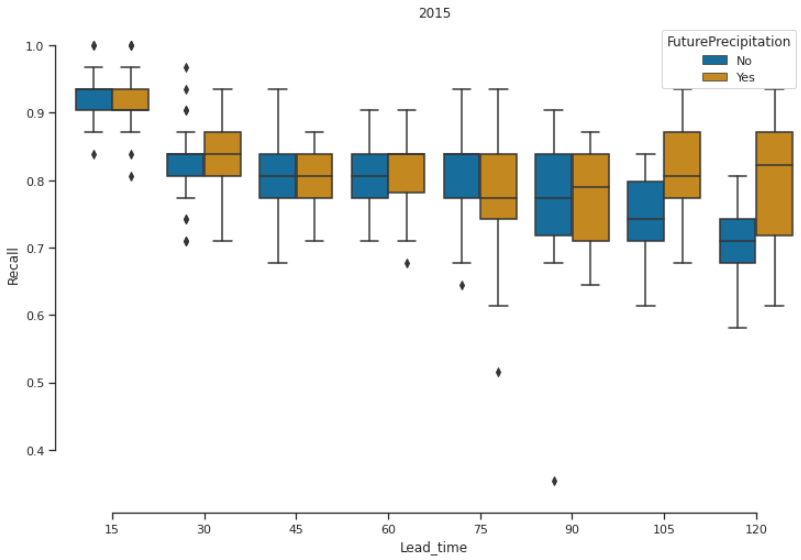

Figure 28: Box plots depicting recall values corresponding to the year 2015 (30 runs).

To get a better understanding of the recall values distributions corresponding to the 30 experimental runs, violin plots were constructed (Figures 29, 30, and 31). It can be clearly seen that models with the future observed rainfall values have higher performance. Particularly, for extended lead times starting from 75 minutes, the increase in predictive performance is significant.

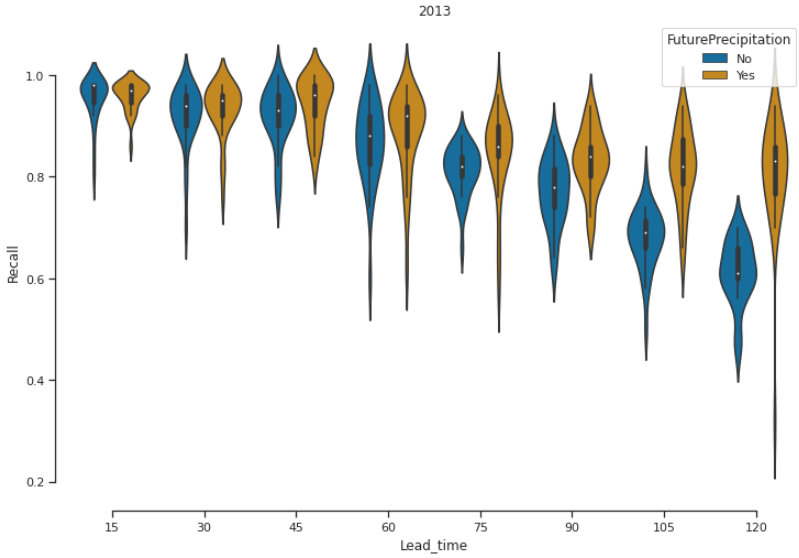

Figure 29: Violin plots corresponding to 30 runs recall values for the year 2013.

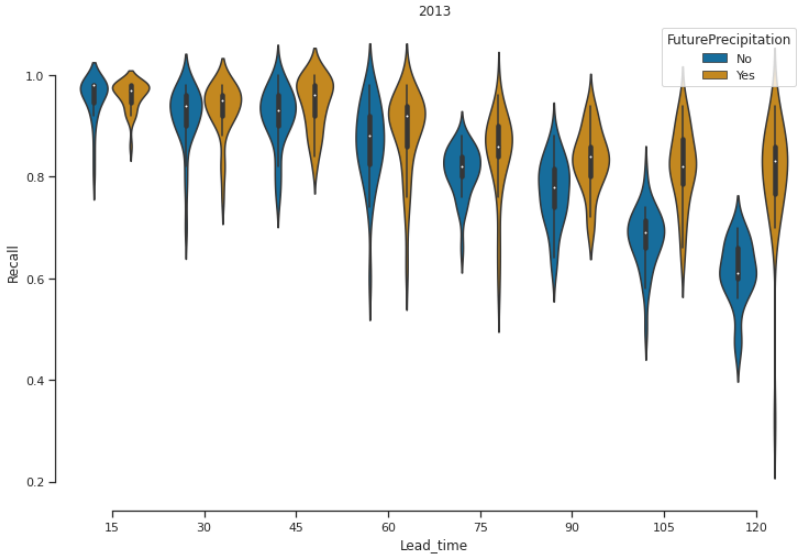

Figure 30: Violin plots corresponding to 30 runs recall values for the year 2014.

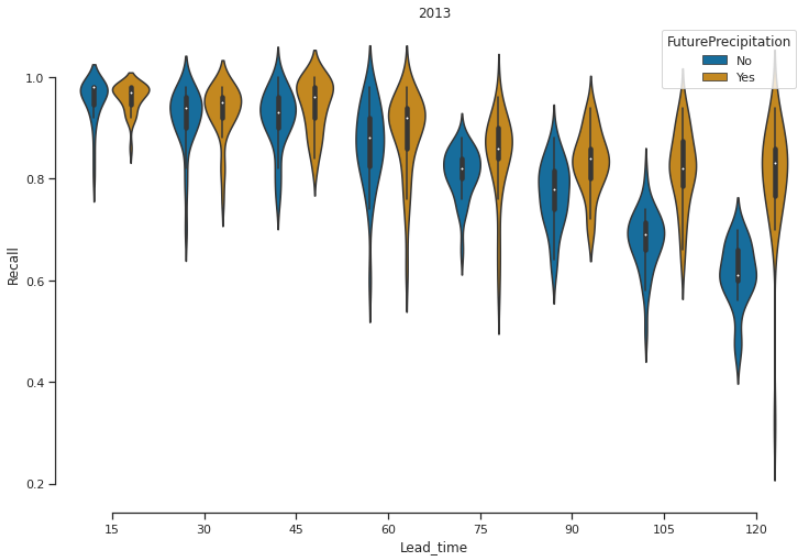

Figure 31: Violin plots corresponding to 30 runs recall values for the year 2015.

Similar findings were obtained based on the F1 score evaluation metric, where an increase in F1 score mean value reached roughly 60%, 20%, and 10% for the years 2013, 2014, and 2015 respectively, for the two hours lead time as illustrated in figures 32, 33, and 34. As far as the precision metric, there was a slight improvement in most of the extended lead times and years. In the worst cases, precision values obtained after introducing precipitation were comparable to those prior to introducing future rainfall values. For clarity reasons, detailed and thorough results corresponding to precision and recall are presented in appendix A.

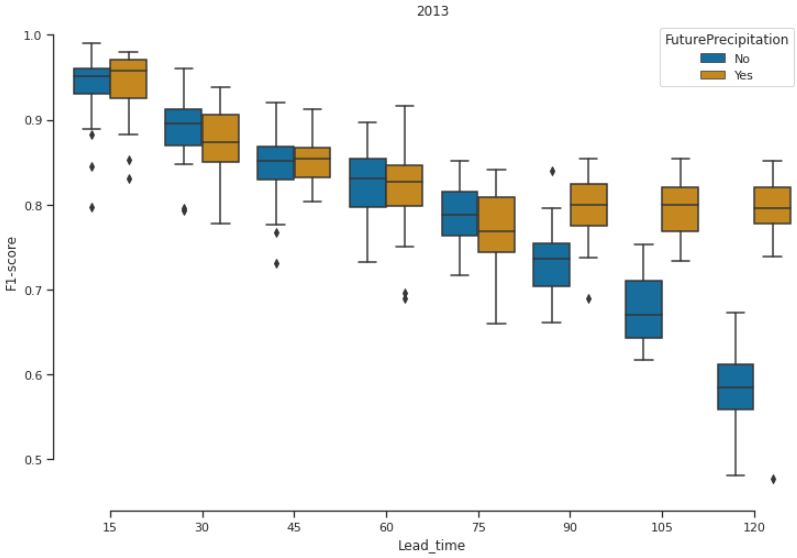

Figure 32: Box plots depicting f1-score values corresponding to the year 2013 (30 runs).

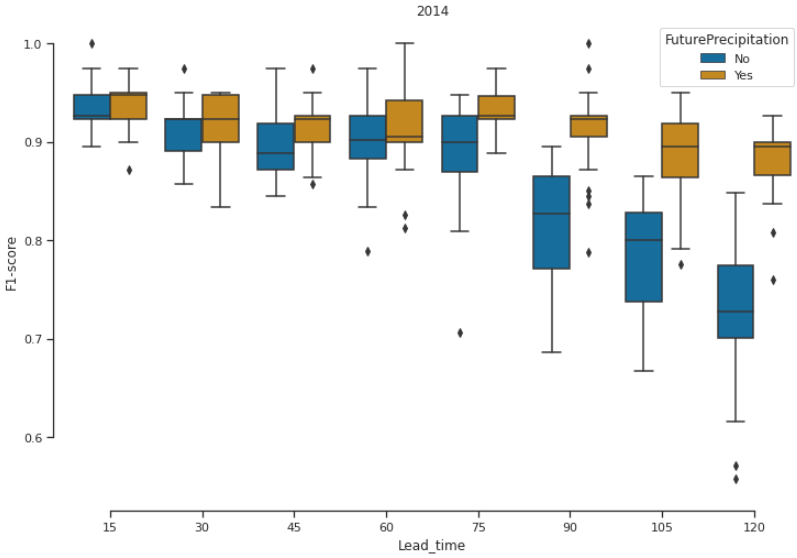

Figure 33: Box plots depicting f1-score values corresponding to the year 2014 (30 runs).

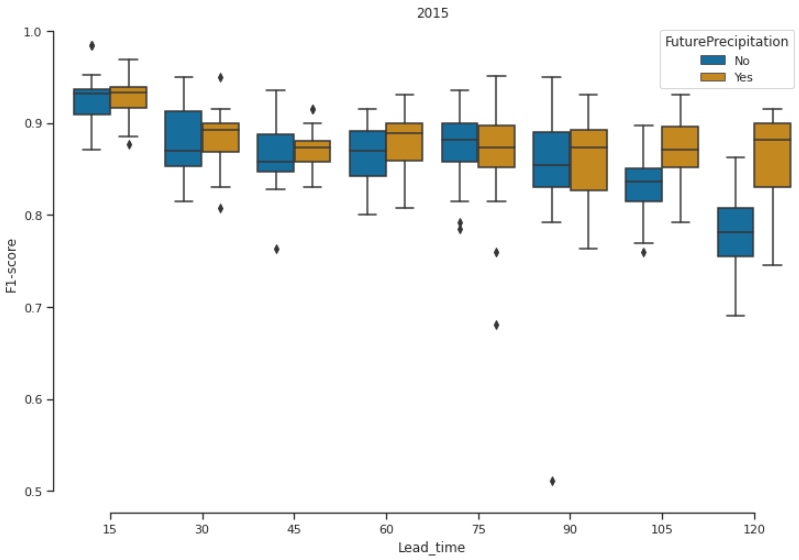

Figure 34: Box plots depicting f1-score values corresponding to the year 2015 (30 runs).

Computational experiments related to studying the uncertainty of predicted rainfall values showed that models' performance varied depending on the type of error (Figures 35 and 36). The error bar lengths represent the mean recall value '+' and '-' one standard deviation, respectively. Figure 35 demonstrates that overestimating future rainfall values increased the flash flood prediction model recall values. Oppositely, underestimation of future precipitations decreased the model performance. We argue that increasing future rainfall magnitude reinforced their signals into the decoder model, which explains the increase in recall. On the other hand, time errors had a less significant effect compared to the magnitude errors, and no consistent pattern was detected. Nevertheless, overall, the introduction of future rainfall values with uncertainty gave better performance than not considering them at all.

The compounding effect of these errors on the flash flood prediction framework is shown in Tables 5.2, 5.3, and 5.4 representing the Recall, Precision, and F1 scores for the combinations of a shift forward time error with magnitude

errors for lead times ranging from 75 to 120 minutes (similar results were obtained for the backward shift) for the years 2013, 2014, and 2015. As expected, the incorporation of precipitation values and time errors decreased the models' performance compared to the observed precipitations and separate errors effect. However, this set of experiments depicts the real-life scenario more closely. Overall, for most of the extended lead times and years, the performance of the model is better than the one without rainfall forecasts at all, except for when the rainfall values are underestimated. However, the performance improvement in terms of recall coincided with a decrease in precision as the number of false-positive occurrences grew. These findings corroborate the importance and benefits of the introduction of rainfall nowcasts for flash flood events prediction. More focus should be put on providing reliable rainfall forecasts, with special attention given to the underestimation of rainfall values which must be reduced.

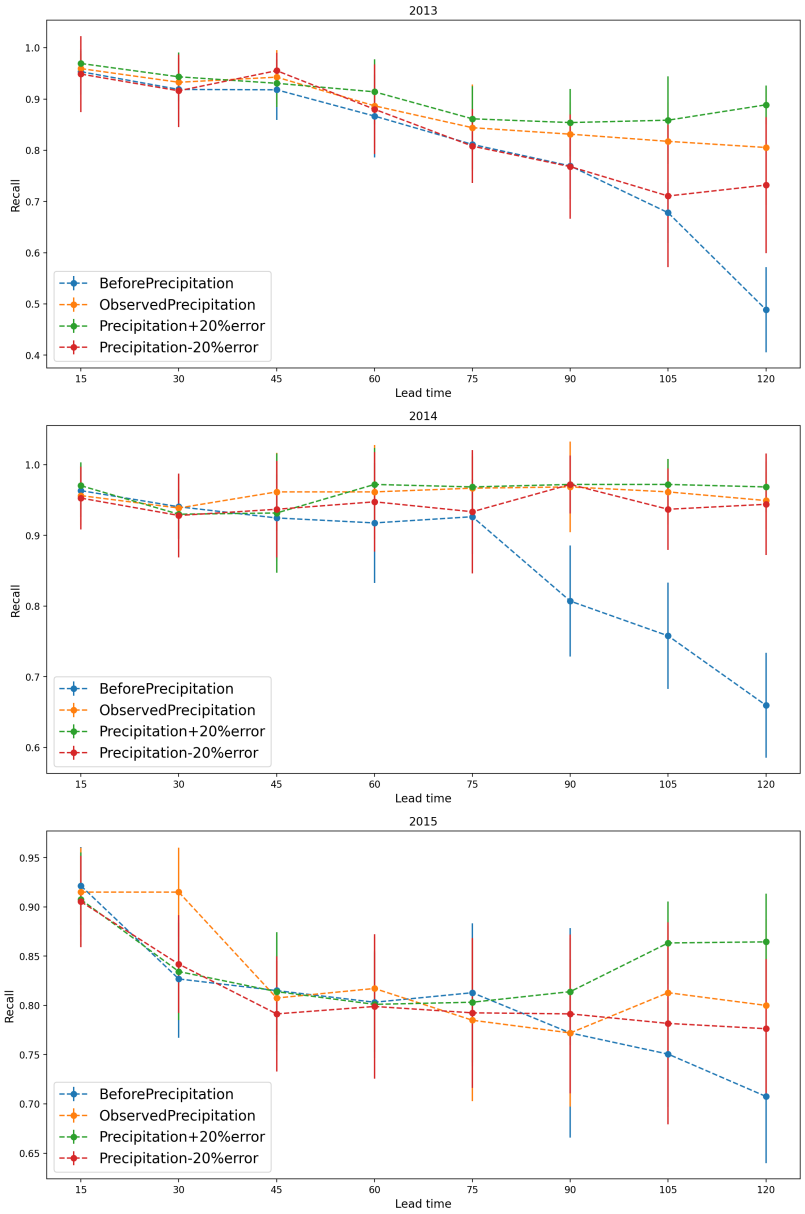

Figure 35: The flash flood model performance in terms of recall after injecting magnitude errors into future rainfall values.

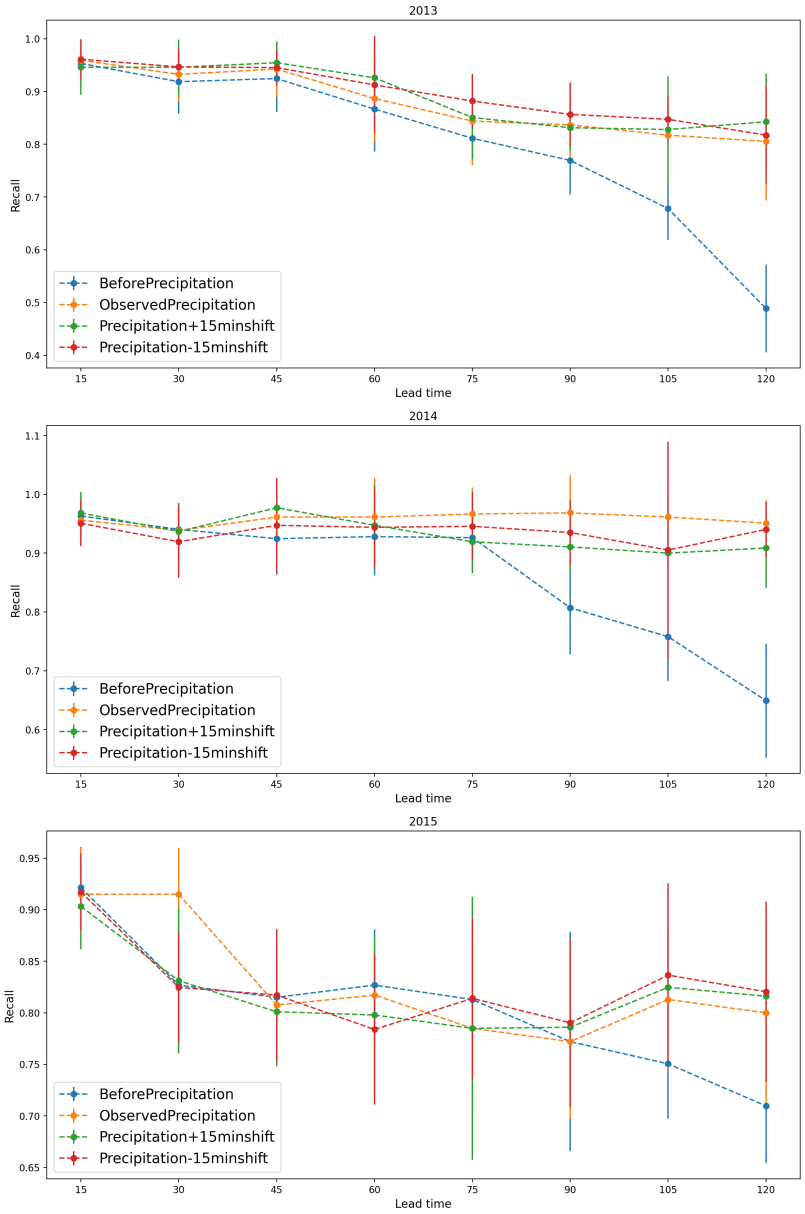

Figure 36: The flash flood model performance in terms of recall after injecting errors in the time of occurrence of rainfall events.

The last set of experiments corresponded to integrating the simulated rainfall forecasts into the flash flood prediction model. Figure 37 shows the mean recall values corresponding to introducing the point estimates generated by the rainfall prediction model. To account for uncertainty, the point estimates generated '+' and '-' two Standard Deviation were as well integrated, and their effect

Table 5.2: The flash flood prediction model performance on 2013 data including future rainfall value with a shift forward time error combined with magnitude errors.

Lead time	**Precision** Mean Max Min SD	**Recall** Mean Max Min SD	**F1** Mean Max Min SD	**Precision** Mean Max Min SD	**Recall** Mean Max Min SD	**F1** Mean Max Min SD
	+20%			-20%		
2013						
75	0.511	0.837	0.601	0.521	0.791	0.599
	0.830	0.960	0.804	0.882	0.900	0.820
	0.170	0.720	0.288	0.1929	0.560	0.295
	0.222	0.059	0.168	0.215	0.073	0.157
90	0.561	0.823	0.645	0.509	0.749	0.570
	0.824	0.920	0.832	0.886	0.920	0.830
	0.234	0.540	0.370	0.183	0.380	0.302
	0.181	0.079	0.138	0.227	0.117	0.167
105	0.525	0.830	0.614	0.634	0.593	0.583
	0.829	0.920	0.792	0.967	0.880	0.800
	0.115	0.600	0.204	0.138	0.180	0.157
	0.205	0.086	0.162	0.240	0.162	0.170
120	0.583	0.839	0.666	0.580	0.625	0.558
	0.871	0.940	0.839	0.871	0.920	0.780
	0.318	0.540	0.462	0.174	0.260	0.258
	0.165	0.101	0.115	0.222	0.176	0.151

Table 5.3: The flash flood prediction model performance on 2014 data including future rainfall values with a shift forward time error combined with magnitude errors.

Lead time	**Precision Mean Max Min SD**	**Recall Mean Max Min SD**	**F1 Mean Max Min SD**	**Precision Mean Max Min SD**	**Recall Mean Max Min SD**	**F1 Mean Max Min SD**
	+20%			-20%		
2014						
75	0.805	0.925	0.857	0.882	0.904	0.888
	0.941	1	0.950	1	1	0.947
	0.613	0.789	0.760	0.633	0.789	0.776
	0.078	0.051	0.042	0.079	0.062	0.041
90	0.758	0.921	0.829	0.819	0.916	0.861
	0.889	1	0.905	0.944	1	0.923
	0.545	0.842	0.692	0.600	0.842	0.735
	0.065	0.043	0.040	0.084	0.045	0.041
105	0.740	0.917	0.817	0.791	0.914	0.845
	0.889	0.947	0.900	0.900	1	0.923
	0.607	0.842	0.723	0.562	0.789	0.706
	0.607	0.842	0.723	0.065	0.051	0.039
120	0.706	0.921	0.796	0.780	0.882	0.824
	0.857	1	0.900	0.867	0.947	0.878
	0.514	0.842	0.667	0.667	0.474	0.581
	0.079	0.043	0.045	0.049	0.098	0.055

Table 5.4: The flash flood prediction model performance on 2015 data including future rainfall values with a shift forward time error combined with magnitude errors.

Lead time	**Precision Mean Max Min SD**	**Recall Mean Max Min SD**	**F1 Mean Max Min SD**	**Precision Mean Max Min SD**	**Recall Mean Max Min SD**	**F1 Mean Max Min SD**
	+20%			-20%		
2015						
75	0.963	0.802	0.873	0.980	0.747	0.843
	1	0.935	0.949	1	0.871	0.915
	0.806	0.645	0.741	0.893	0.452	0.622
	0.049	0.065	0.043	0.030	0.099	0.068
90	0.878	0.838	0.855	0.973	0.729	0.830
	1	0.935	0.912	1	0.871	0.912
	0.722	0.742	0.762	0.711	0.548	0.708
	0.072	0.044	0.039	0.056	0.073	0.046
105	0.754	0.859	0.801	0.963	0.662	0.782
	0.931	0.903	0.900	1	0.806	0.893
	0.560	0.677	0.656	0.750	0.548	0.708
	0.073	0.044	0.048	0.0523	0.069	0.046
120	0.648	0.851	0.733	0.980	0.546	0.697
	0.793	0.903	0.818	1	0.677	0.808
	0.475	0.742	0.622	0.895	0.387	0.558
	0.071	0.045	0.048	0.033	0.088	0.070

studied. Overall, the observed precipitation did much better. Compared to the model without future rainfall values, simulated rainfall predictions improved the models' performance on the two-hour lead time and gave a slight improvement on the 75 minutes lead time. Similar to the basic uncertainty estimates experiments, overestimating rainfall predictions delivered high recall values while underestimating of precipitation decreased recall values considerably.

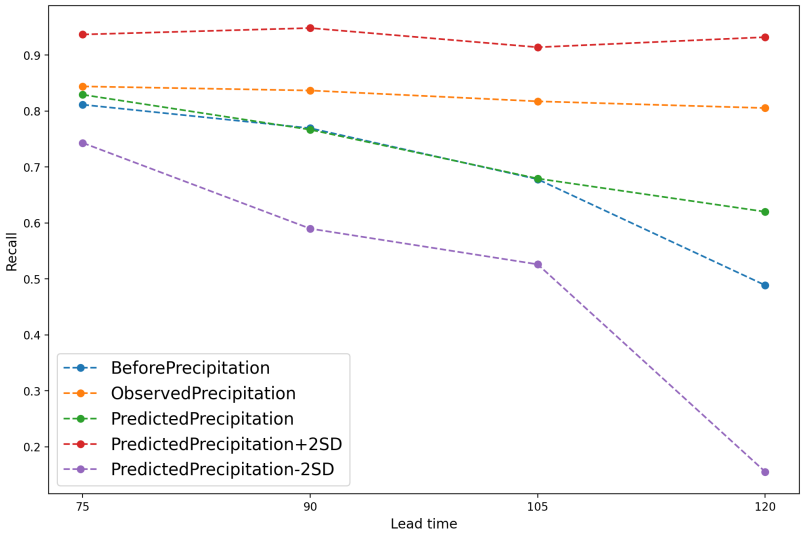

Figure 37: The flash flood model performance after introducing the data-driven rainfall nowcasts.

Chapter 6

Conclusion

This thesis investigated the effect of rainfall forecasts on the predictive ability of a data-driven model for flash floods prediction. It consisted of three major parts. Firstly, a data-driven rainfall nowcasting model was built. Secondly, the developed model was integrated into a flash flood forecasting framework. In the third part, the effect of rainfall nowcasts, including their uncertainty, was studied. It is important to mention that flash floods are characterized by the very rapid changes in the magnitude of investigated variables making modeling such events extremely challenging.

The dataset consisted of rainfall and water level measurements provided by TRCA. These data were collected on a watershed in Toronto, Ontario, Canada, with a response time of approximately 30 minutes. The data correspond to the warm period of four years: 2013, 2014, 2015, and 2016 which had distinct hydrological characteristics.

We followed the KDD framework steps to build the applications. For the rainfall modeling problem we investigated two approaches: a univariate (data obtained from a single sensor was used for prediction) and a multivariate ap-

proach (data gathered from neighboring sensors was used to predict rainfall at the target location). In general, we found that for rainfall nowcasting using univariate data, the LSTM model outperformed the ARIMA and naïve models with the exception of the year 2015, where major discrepancies in the rainfall distribution were observed. Nonetheless, all the models were unable to accurately predict extreme values and performed similarly to the naïve model. Computational experiments showed that integration of data from several sensors brought major improvement to models' performance depending on the hydrological characteristics of the year and the location of sensors. More specifically, using lagged values from sensors that coincided with the dominating direction of the air flow enhanced models' predictive ability considerably. Therefore, more focus should be placed on data representativeness and richness.

In the second part, a flash flood prediction framework was proposed. It was built based on LSTM-based DL algorithm and integration of precipitation nowcasts into the predictive models. The developed framework allows for the investigation of the effect of precipitation nowcasts and their uncertainty on flash flood events prediction models.

This effect was examined in the third part of the study. We found that introduction of rainfall nowcasts significantly enhanced flash flood predictability both in terms of recall and F1-score. Experiments showed that introducing the data-driven rainfall nowcasts enhanced the model performance for two hours lead-time on the data observed on the watershed. We found that underestimating precipitation predictions had the biggest negative effect on the flash flood prediction framework performance.

Recommendations and future work

Based on the obtained results, we recommend the integration of rainfall forecasts for flash flood modeling. However, these forecasts need to be within an acceptable margin of error. For that, future efforts should focus on enhancing rainfall nowcasting models. We suggest the following steps and research directions:

- Enhance the rainfall nowcasting models' performance through integrating other meteorological variables.
- Consider the ensembling methodology to improve the predictive ability of both the rainfall and flash flood prediction framework. Ensembling ANN-based models and Tree-based models is a promising direction.
- Explore methods to deploy the developed models into the cloud and build their APIs.

Bibliography

[1] S. J. Russell and P. Norvig, *Artificial Intelligence: A modern approach*. Pearson Education Limited, 2013, p. 736–748.

[2] D. Hughes and N. Correll, "Distributed machine learning in materials that couple sensing, actuation, computation and communication," *ArXiv*, 2016.

[3] F. Chollet, *Deep learning with python*. Manning Publications, 2021.

[4] M. Kana, "Uncertainty in deep learning. how to measure?" May 2020. [Online]. Available: https://towardsdatascience.com/my-deep-learning-model-says-sorry-i-dont-know-the-answer-that-s-absolutely-ok-50ffa562cb0b

[5] N. Srivastava, G. Hinton, A. Krizhevsky, I. Sutskever, and R. Salakhutdinov, "Dropout: A simple way to prevent neural networks from overfitting," *Journal of Machine Learning Research*, vol. 15, no. 56, pp. 1929–1958, 2014. [Online]. Available: http://jmlr.org/papers/v15/srivastava14a.html

[6] A. Brando, J. A. Rodríguez-Serrano, M. Ciprian, R. Maestre, and J. Vitrià, "Uncertainty modelling in deep networks: Forecasting short and noisy series," *ArXiv*, vol. abs/1807.09011, 2018.

[7] V. T. Chow, *Applied Hydrology*. McGraw-Hill, 1988.

[8] L. Rokach and O. Maimon, *Data Mining with Decision Trees*. WORLD SCIENTIFIC, dec 2013.

[9] TRCA, "Etobicoke creek watershed report card 2018," 2018. [Online]. Available: https://reportcard.trca.ca/watershed-report-cards/etobicoke-creek/

[10] Z. Cui and Y. Wang, "Deep stacked bidirectional and unidirectional lstm recurrent neural network for network-wide traffic speed prediction," 2017.

[11] D. Martínez, A. Fernández Llamas, P. Hernández, S. Cristóbal, F. Schwaiger, J. Nuñez, and J. Ruiz, "Forecasting unstable approaches with boosting frameworks and lstm networks," Dec 2019.

[12] W. Junker, "Flash flooding: A lesser known, but leading weather killer," Dec 2021. [Online]. Available: https://www.washingtonpost.com/news/capital-weather-gang/wp/2013/07/11/flash-flooding-a-lesser-known-but-leading-weather-killer/

[13] S. Kevin, *Flash floods: forecasting and warning*. Springer, 2015.

[14] C. Mills, "Toronto's july flood listed as ontario's most costly natural disaster," Aug 2013. [Online]. Available: https://www.thestar.com/business/2013/08/14/july_flood_ontarios_most_costly_natural_disaster.html

[15] C. Hu, Q. Wu, H. Li, S. Jian, N. Li, and Z. Lou, "Deep learning with a long short-term memory networks approach for rainfall-runoff simulation," *Water*, vol. 10, no. 11, p. 1543, oct 2018.

[16] F. Kratzert, D. Klotz, C. Brenner, K. Schulz, and M. Herrnegger, "Rainfall–runoff modelling using long short-term memory (LSTM) networks," *Hydrology and Earth System Sciences*, vol. 22, no. 11, pp. 6005–6022, nov 2018.

[17] P. Tao, S. Tie-Yuan, Y. Zhi-Yuan, and W. Jun-Chao, "Application of quantitative precipitation forecasting and precipitation ensemble prediction for hydrological forecasting," *Proceedings of the International Association of Hydrological Sciences*, vol. 368, pp. 96–101, may 2015.

[18] F. Hussain, R.-S. Wu, and J.-X. Wang, "Comparative study of very short-term flood forecasting using physics-based numerical model and data-driven prediction model," *Natural Hazards*, vol. 107, no. 1, pp. 249–284, feb 2021.

[19] T. Song, W. Ding, J. Wu, H. Liu, H. Zhou, and J. Chu, "Flash flood forecasting based on long short-term memory networks," *Water*, vol. 12, no. 1, p. 109, dec 2019.

[20] R. Mhedhbi and M. G. Erechtchoukova, "Short term predictions of flash floods in highly urbanized watersheds," Bachelor's thesis, Higher Institute of Applied Sciences and Technology of Sousse, Sousse University, 2019.

[21] A. D. L. Zanchetta and P. Coulibaly, "Recent advances in real-time pluvial flash flood forecasting," *Water*, vol. 12, no. 2, 2020. [Online]. Available: https://www.mdpi.com/2073-4441/12/2/570

[22] M. G. Erechtchoukova, P. A. Khaiter, and S. Saffarpour, "Short-term predictions of hydrological events on an urbanized watershed using super-

vised classification," *Water Resources Management*, vol. 30, no. 12, pp. 4329–4343, jul 2016.

[23] E. Alpaydin, *Introduction to machine learning*. The MIT Press, 2014.

[24] O. I. Abiodun, A. Jantan, A. E. Omolara, K. V. Dada, N. A. Mohamed, and H. Arshad, "State-of-the-art in artificial neural network applications: A survey," *Heliyon*, vol. 4, no. 11, p. e00938, 2018. [Online]. Available: https://www.sciencedirect.com/science/article/pii/S2405844018332067

[25] A. Kendall and Y. Gal, "What uncertainties do we need in bayesian deep learning for computer vision?" *Advances in neural information processing systems*, vol. 30, 2017.

[26] M. Abdar, F. Pourpanah, S. Hussain, D. Rezazadegan, L. Liu, M. Ghavamzadeh, P. Fieguth, X. Cao, A. Khosravi, U. R. Acharya, V. Makarenkov, and S. Nahavandi, "A review of uncertainty quantification in deep learning: Techniques, applications and challenges," *Information Fusion*, vol. 76, pp. 243–297, dec 2021.

[27] J. Gawlikowski, C. R. N. Tassi, M. Ali, J. Lee, M. Humt, J. Feng, A. Kruspe, R. Triebel, P. Jung, R. Roscher, M. Shahzad, W. Yang, R. Bamler, and X. X. Zhu, "A survey of uncertainty in deep neural networks."

[28] M. Abdar, F. Pourpanah, S. Hussain, D. Rezazadegan, L. Liu, M. Ghavamzadeh, P. Fieguth, X. Cao, A. Khosravi, U. R. Acharya, V. Makarenkov, and S. Nahavandi, "A review of uncertainty quantification in deep learning: Techniques, applications and challenges," *Information Fusion*, vol. 76, pp. 243–297, dec 2021.

[29] J. Gawlikowski, C. R. N. Tassi, M. Ali, J. Lee, M. Humt, J. Feng, A. Kruspe, R. Triebel, P. Jung, R. Roscher, M. Shahzad, W. Yang, R. Bamler, and X. X. Zhu, "A survey of uncertainty in deep neural networks."

[30] Y. Gal and Z. Ghahramani, "A theoretically grounded application of dropout in recurrent neural networks," *Advances in neural information processing systems*, vol. 29, 2016.

[31] G. Yarin and G. Zoubin, "Dropout as a bayesian approximation: Representing model uncertainty in deep learning," in *international conference on machine learning*. PMLR, 2016, pp. 1050–1059.

[32] J. Davis, J. Zhu, J. Oldfather, S. MacDonald, M. Trzaskowski, and M. Kelsen, "Quantifying uncertainty in deep learning systems," Aws Perspective Guidance, Aug. 2020.

[33] B. Lakshminarayanan, A. Pritzel, and C. Blundell, "Simple and scalable predictive uncertainty estimation using deep ensembles," *Advances in neural information processing systems*, vol. 30, 2017.

[34] L. Zhu and N. Laptev, "Deep and confident prediction for time series at uber."

[35] D. Althoff, L. N. Rodrigues, and H. C. Bazame, "Uncertainty quantification for hydrological models based on neural networks: the dropout ensemble," *Stochastic Environmental Research and Risk Assessment*, vol. 35, no. 5, pp. 1051–1067, feb 2021.

[36] L. Xu, N. Chen, and C. Yang, "Quantifying the uncertainty of precipitation forecasting using probabilistic deep learning," *Hydrology and Earth System Sciences Discussions*, pp. 1–27, 2021.

[37] H. Taras, "Quantifying uncertainty in deep learning," Master's thesis, Harvard College, 2020.

[38] "Flood types." [Online]. Available: https://www.nssl.noaa.gov/education/svrwx101/floods/types/

[39] C. A. Doswell III, *Flooding*. Academic Press, 2003.

[40] P. Wilderer and S. DP. Elsevier Science, 2011, p. 435–457.

[41] M. G. Erechtchoukova and P. A. Khaiter, "The effect of data granularity on prediction of extreme hydrological events in highly urbanized watersheds: A supervised classification approach," vol. 96, pp. 232–238, oct 2017.

[42] Y. Liu, H. Wang, W. Feng, and H. Huang, "Short term real-time rolling forecast of urban river water levels based on LSTM: A case study in fuzhou city, china," *International Journal of Environmental Research and Public Health*, vol. 18, no. 17, p. 9287, sep 2021.

[43] F. Granata, R. Gargano, and G. de Marinis, "Support vector regression for rainfall-runoff modeling in urban drainage: A comparison with the EPA's storm water management model," *Water*, vol. 8, no. 3, p. 69, feb 2016.

[44] C. R. Rivero, Y. Tupac, J. Pucheta, G. Juarez, L. Franco, and P. Otano, "Time-series prediction with BEMCA approach: Application to short

rainfall series," in *2017 IEEE Latin American Conference on Computational Intelligence (LA-CCI)*. IEEE, nov 2017.

[45] X. Huang, Y. Li, Z. Tian, Q. Ye, Q. Ke, D. Fan, G. Mao, A. Chen, and J. Liu, "Evaluation of short-term streamflow prediction methods in urban river basins," *Physics and Chemistry of the Earth, Parts A/B/C*, vol. 123, p. 103027, oct 2021.

[46] J. Yan, J. Jin, F. Chen, G. Yu, H. Yin, and W. Wang, "Urban flash flood forecast using support vector machine and numerical simulation," *Journal of Hydroinformatics*, vol. 20, no. 1, pp. 221–231, nov 2017.

[47] Z. Cui, Y. Zhou, S. Guo, J. Wang, H. Ba, and S. He, "A novel hybrid XAJ-LSTM model for multi-step-ahead flood forecasting," *Hydrology Research*, vol. 52, no. 6, pp. 1436–1454, jun 2021.

[48] B. Alizadeh, A. G. Bafti, H. Kamangir, Y. Zhang, D. B. Wright, and K. J. Franz, "A novel attention-based LSTM cell post-processor coupled with bayesian optimization for streamflow prediction," *Journal of Hydrology*, vol. 601, p. 126526, oct 2021.

[49] T. A. Khan, Z. Shahid, M. Alam, M. Su'ud, and K. Kadir, "Early flood risk assessment using machine learning: A comparative study of SVM, q-SVM, k-NN and LDA," in *2019 13th International Conference on Mathematics, Actuarial Science, Computer Science and Statistics (MACS)*. IEEE, dec 2019.

[50] P. Ghasemigoudarzi, W. Huang, and O. D. Silva, "Detecting floods caused by tropical cyclone using CYGNSS data," in *2020 IEEE Interna-*

tional Conference on Multisensor Fusion and Integration for Intelligent Systems (MFI). IEEE, sep 2020.

[51] G. Suddul, K. Dookhitram, G. Bekaroo, and N. Shankhur, "An evolutionary MultiLayer perceptron algorithm for real time river flood prediction," in *2020 Zooming Innovation in Consumer Technologies Conference (ZINC)*. IEEE, may 2020.

[52] Q. Ke, X. Tian, J. Bricker, Z. Tian, G. Guan, H. Cai, X. Huang, H. Yang, and J. Liu, "Urban pluvial flooding prediction by machine learning approaches – a case study of shenzhen city, china," *Advances in Water Resources*, vol. 145, p. 103719, nov 2020.

[53] D. Han, T. Kwong, and S. Li, "Uncertainties in real-time flood forecasting with neural networks," *Hydrological Processes*, vol. 21, no. 2, pp. 223–228, 2007.

[54] J. Wu, H. Liu, G. Wei, T. Song, C. Zhang, and H. Zhou, "Flash flood forecasting using support vector regression model in a small mountainous catchment," *Water*, vol. 11, no. 7, p. 1327, jun 2019.

[55] Y. Tikhamarine, D. Souag-Gamane, A. N. Ahmed, S. S. Sammen, O. Kisi, Y. F. Huang, and A. El-Shafie, "Rainfall-runoff modelling using improved machine learning methods: Harris hawks optimizer vs. particle swarm optimization," *Journal of Hydrology*, vol. 589, p. 125133, oct 2020.

[56] G. Kan, K. Liang, H. Yu, B. Sun, L. Ding, J. Li, X. He, and C. Shen, "Hybrid machine learning hydrological model for flood forecast purpose," *Open Geosciences*, vol. 12, no. 1, pp. 813–820, jan 2020.

[57] S. Zhang, L. Lu, J. Yu, and H. Zhou, "Short-term water level prediction using different artificial intelligent models," in *2016 Fifth International Conference on Agro-Geoinformatics (Agro-Geoinformatics)*. IEEE, jul 2016.

[58] Pichaid, "Flood forecasting using artificial neural networks," Ph.D. dissertation, Balkema, 2003.

[59] F.-J. Chang, P.-A. Chen, Y.-R. Lu, E. Huang, and K.-Y. Chang, "Real-time multi-step-ahead water level forecasting by recurrent neural networks for urban flood control," *Journal of Hydrology*, vol. 517, pp. 836–846, sep 2014.

[60] M. Sit and I. Demir, "Decentralized flood forecasting using deep neural networks."

[61] Y. Ding, Y. Zhu, J. Feng, P. Zhang, and Z. Cheng, "Interpretable spatio-temporal attention LSTM model for flood forecasting," *Neurocomputing*, vol. 403, pp. 348–359, aug 2020.

[62] N. Lv, X. Liang, C. Chen, Y. Zhou, J. Li, H. Wei, and H. Wang, "A long short-term memory cyclic model with mutual information for hydrology forecasting: A case study in the xixian basin," *Advances in Water Resources*, vol. 141, p. 103622, jul 2020.

[63] M. Fu, T. Fan, Z. Ding, S. Q. Salih, N. Al-Ansari, and Z. M. Yaseen, "Deep learning data-intelligence model based on adjusted forecasting window scale: Application in daily streamflow simulation," *IEEE Access*, vol. 8, pp. 32 632–32 651, 2020.

[64] M. Rahimzad, A. M. Nia, H. Zolfonoon, J. Soltani, A. D. Mehr, and H.-H. Kwon, "Performance comparison of an LSTM-based deep learning model versus conventional machine learning algorithms for streamflow forecasting," *Water Resources Management*, vol. 35, no. 12, pp. 4167–4187, aug 2021.

[65] S. Dazzi, R. Vacondio, and P. Mignosa, "Flood stage forecasting using machine-learning methods: A case study on the parma river (italy)," *Water*, vol. 13, no. 12, p. 1612, jun 2021.

[66] C. E. Brendel, R. L. Dymond, and M. F. Aguilar, "Integration of quantitative precipitation forecasts with real-time hydrology and hydraulics modeling towards probabilistic forecasting of urban flooding," *Environmental Modelling & Software*, vol. 134, p. 104864, dec 2020.

[67] P. Tao, S. Tie-Yuan, Y. Zhi-Yuan, and W. Jun-Chao, "Application of quantitative precipitation forecasting and precipitation ensemble prediction for hydrological forecasting," *Proceedings of the International Association of Hydrological Sciences*, vol. 368, pp. 96–101, may 2015.

[68] Z. Xiang, J. Yan, and I. Demir, "A rainfall-runoff model with LSTM-based sequence-to-sequence learning," *Water Resources Research*, vol. 56, no. 1, jan 2020.

[69] Z. Xiang and I. Demir, "Distributed long-term hourly streamflow predictions using deep learning – a case study for state of iowa," *Environmental Modelling & Software*, vol. 131, p. 104761, sep 2020.

[70] T. Song, W. Ding, H. Liu, J. Wu, H. Zhou, and J. Chu, "Uncertainty quantification in machine learning modeling for multi-step time series

forecasting: Example of recurrent neural networks in discharge simulations," *Water*, vol. 12, no. 3, p. 912, mar 2020.

[71] H. Yin, X. Zhang, F. Wang, Y. Zhang, R. Xia, and J. Jin, "Rainfall-runoff modeling using LSTM-based multi-state-vector sequence-to-sequence model," *Journal of Hydrology*, vol. 598, p. 126378, jul 2021.

[72] *Manual on the Global Data-processing and Forecasting System*. World Meteorological Organization, 2010.

[73] R. B. Stall, "Predictability and scales of motion," *Bulletin of the American Meteorological Society*, vol. 66, no. 4, pp. 432–436, Apr. 1985. [Online]. Available: https://doi.org/10.1175/1520-0477-66.4.432

[74] S. Agrawal, L. Barrington, C. Bromberg, J. Burge, C. Gazen, and J. Hickey, "Machine learning for precipitation nowcasting from radar images," *arXiv preprint arXiv:1912.12132*, 2019.

[75] A. Akbari Asanjan, "An advanced deep learning framework for short-term precipitation forecasting fromsatellite information," Ph.D. dissertation, University of California, 2019.

[76] D. Nerini, "Ensemble precipitation nowcasting: limits to prediction, localization and seamless blending," Ph.D. dissertation, ETH Zurich, 2019.

[77] M. A. Nayak and S. Ghosh, "Prediction of extreme rainfall event using weather pattern recognition and support vector machine classifier," *Theoretical and Applied Climatology*, vol. 114, no. 3-4, pp. 583–603, mar 2013.

[78] R. E. Caraka, R. C. Chen, B. D. Supatmanto, Arnita, M. Tahmid, and T. Toharudin, "Employing moving average long short term memory for

predicting rainfall," in *2019 International Conference on Technologies and Applications of Artificial Intelligence (TAAI)*. IEEE, nov 2019.

[79] C. Wang, P. Wang, D. Wang, J. Hou, and B. Xue, "Nowcasting multicell short-term intense precipitation using graph models and random forests," *Monthly Weather Review*, vol. 148, no. 11, pp. 4453–4466, nov 2020.

[80] P. Zhang, Y. Jia, J. Gao, W. Song, and H. Leung, "Short-term rainfall forecasting using multi-layer perceptron," *IEEE Transactions on Big Data*, vol. 6, no. 1, pp. 93–106, mar 2020.

[81] H. Xie, L. Wu, W. Xie, Q. Lin, M. Liu, and Y. Lin, "Improving ECMWF short-term intensive rainfall forecasts using generative adversarial nets and deep belief networks," *Atmospheric Research*, vol. 249, p. 105281, feb 2021.

[82] A. Tsonis and G. Austin, "An evaluation of extrapolation techniques for the short-term prediction of rain amounts," *Atmosphere-Ocean*, vol. 19, no. 1, pp. 54–65, mar 1981.

[83] A. Bellon and G. Austin, "The accuracy of short-term radar rainfall forecasts," *Journal of Hydrology*, vol. 70, no. 1-4, pp. 35–49, feb 1984.

[84] J. Olsson, L. Simonsson, and M. Ridal, "Rainfall nowcasting: predictability of short-term extremes in sweden," *Urban Water Journal*, vol. 11, no. 7, pp. 605–615, nov 2013.

[85] D. RanjanNayak, A. Mahapatra, and P. Mishra, "A survey on rainfall prediction using artificial neural network," *International Journal of Computer Applications*, vol. 72, no. 16, pp. 32–40, jun 2013.

[86] B. K. Rani and A. Govardhan, "Rainfall prediction using data mining techniques - a survey," in *Computer Science & Information Technology (CS & IT)*. Academy & Industry Research Collaboration Center (AIRCC), dec 2013.

[87] A. S. Abdullah, B. N. Ruchjana, I. G. N. M. Jaya, and Soemartini, "Comparison of SARIMA and SVM model for rainfall forecasting in bogor city, indonesia," *Journal of Physics: Conference Series*, vol. 1722, p. 012061, jan 2021.

[88] C. Wu, K. Chau, and C. Fan, "Prediction of rainfall time series using modular artificial neural networks coupled with data-preprocessing techniques," *Journal of Hydrology*, vol. 389, no. 1-2, pp. 146–167, jul 2010.

[89] A. V. Kulkarni, "Machine learning applied to build 'nowcasting' models to predict irish rainfall," Master's thesis, School of Mathematics and Statistics, University College Dublin, 2019.

[90] S. Shalev-Shwartz and S. Ben-David, *Understanding Machine Learning - From Theory to Algorithms.* Cambridge University Press, 2014.

[91] D. M. Casas, J. Á. T. González, J. E. A. Rodríguez, and J. V. Pet, "Using data-mining for short-term rainfall forecasting," in *Distributed Computing, Artificial Intelligence, Bioinformatics, Soft Computing, and Ambient Assisted Living*. Springer Berlin Heidelberg, 2009, pp. 487–490.

[92] T. Mandal and V. Jothiprakash, "Short-term rainfall prediction using ANN and MT techniques," *ISH Journal of Hydraulic Engineering*, vol. 18, no. 1, pp. 20–26, mar 2012.

[93] M. S. Balamurugan and R. Manojkumar, "Study of short term rain forecasting using machine learning based approach," *Wireless Networks*, oct 2019.

[94] P. Vogel, P. Knippertz, T. Gneiting, A. H. Fink, M. Klar, and A. Schlueter, "Statistical forecasts for the occurrence of precipitation outperform global models over northern tropical africa," *Geophysical Research Letters*, vol. 48, no. 3, feb 2021.

[95] P. Verma and S. Chakraborty, "Now casting of orographic rain rate using ARIMA and ANN model," in *2020 URSI Regional Conference on Radio Science (URSI-RCRS)*. IEEE, feb 2020.

[96] E. Toth, A. Brath, and A. Montanari, "Comparison of short-term rainfall prediction models for real-time flood forecasting," *Journal of Hydrology*, vol. 239, no. 1-4, pp. 132–147, dec 2000.

[97] Y. Yao, L. Shan, and Q. Zhao, "Establishing a method of short-term rainfall forecasting based on GNSS-derived PWV and its application," *Scientific Reports*, vol. 7, no. 1, sep 2017.

[98] P. Krammer, M. Kvassay, O. Habala, and L. Hluchy, "Short-term rainfall estimation by machine learning methods," in *2019 IEEE 15th International Scientific Conference on Informatics*. IEEE, nov 2019.

[99] H. R. Maier and G. C. Dandy, "Neural networks for the prediction and forecasting of water resources variables: a review of modelling issues and applications," *Environmental Modelling & Software*, vol. 15, no. 1, pp. 101–124, jan 2000.

[100] M. P. Darji, V. K. Dabhi, and H. B. Prajapati, "Rainfall forecasting using neural network: A survey," in *2015 International Conference on Advances in Computer Engineering and Applications*. IEEE, mar 2015.

[101] K. Abhishek, A. Kumar, R. Ranjan, and S. Kumar, "A rainfall prediction model using artificial neural network," in *2012 IEEE Control and System Graduate Research Colloquium*. IEEE, jul 2012.

[102] K. Pradip, P. Kumar, and S. Manoj, "Rainfall forecasting using artificial neural network (ann) and adaptive neuro-fuzzy inference system (anfis) models," *International Journal of Agriculture Sciences*, vol. 10, pp. 6153–6159, 05 2018.

[103] S. Lin, J. Li, L. Zhang, and Y. Lu, "Precipitation prediction in shenzhen city based on WNN," in *2017 14th International Computer Conference on Wavelet Active Media Technology and Information Processing (ICCWAMTIP)*. IEEE, dec 2017.

[104] N. Mishra, H. K. Soni, S. Sharma, and A. K. Upadhyay, "Development and analysis of artificial neural network models for rainfall prediction by using time-series data," *International Journal of Intelligent Systems and Applications*, vol. 10, no. 1, pp. 16–23, jan 2018.

[105] R. J. Kuligowski and A. P. Barros, "Experiments in short-term precipitation forecasting using artificial neural networks," *Monthly Weather Review*, vol. 126, no. 2, pp. 470–482, feb 1998.

[106] N. Q. Hung, M. S. Babel, S. Weesakul, and N. K. Tripathi, "An artificial neural network model for rainfall forecasting in bangkok, thailand," *Hy-*

drology and Earth System Sciences, vol. 13, no. 8, pp. 1413–1425, aug 2009.

[107] M. Nasseri, K. Asghari, and M. Abedini, "Optimized scenario for rainfall forecasting using genetic algorithm coupled with artificial neural network," *Expert Systems with Applications*, vol. 35, no. 3, pp. 1415–1421, oct 2008.

[108] S. Poornima and M. Pushpalatha, "Prediction of rainfall using intensified LSTM based recurrent neural network with weighted linear units," *Atmosphere*, vol. 10, no. 11, p. 668, oct 2019.

[109] A. S. Srinivas, R. Somula, G. K., A. Saxena, and P. R. A., "Estimating rainfall using machine learning strategies based on weather radar data," *International Journal of Communication Systems*, vol. 33, no. 13, p. e3999, jun 2019.

[110] N. B. M. Khairudin, N. B. Mustapha, T. N. B. M. Aris, and M. B. Zolkepli, "Comparison of machine learning models for rainfall forecasting," in *2020 International Conference on Computer Science and Its Application in Agriculture (ICOSICA)*. IEEE, sep 2020.

[111] Z. Wang, C. Ye, and Y. Wang, "Prediction of short-term precipitation in qinghai lake based on BiLSTM-attention method," in *Proceedings of the 2019 8th International Conference on Computing and Pattern Recognition*. ACM, oct 2019.

[112] M. Qiu, P. Zhao, K. Zhang, J. Huang, X. Shi, X. Wang, and W. Chu, "A short-term rainfall prediction model using multi-task convolutional neu-

ral networks," in *2017 IEEE International Conference on Data Mining (ICDM)*. IEEE, nov 2017.

[113] V. Lebedev, V. Ivashkin, I. Rudenko, A. Ganshin, A. Molchanov, S. Ovcharenko, R. Grokhovetskiy, I. Bushmarinov, and D. Solomentsev, "Precipitation nowcasting with satellite imagery," in *Proceedings of the 25th ACM SIGKDD International Conference on Knowledge Discovery & Data Mining*. ACM, jul 2019.

[114] M. Xue, R. Hang, Q. Liu, X.-T. Yuan, and X. Lu, "CNN-based near-real-time precipitation estimation from fengyun-2 satellite over xinjiang, china," *Atmospheric Research*, vol. 250, p. 105337, mar 2021.

[115] C.-C. Wei and C.-C. Hsu, "Real-time rainfall forecasts based on radar reflectivity during typhoons: Case study in southeastern taiwan," *Sensors*, vol. 21, no. 4, p. 1421, feb 2021.

[116] G. Franch, D. Nerini, M. Pendesini, L. Coviello, G. Jurman, and C. Furlanello, "Precipitation nowcasting with orographic enhanced stacked generalization: Improving deep learning predictions on extreme events," *Atmosphere*, vol. 11, no. 3, p. 267, mar 2020.

[117] A. A. Asanjan, T. Yang, K. Hsu, S. Sorooshian, J. Lin, and Q. Peng, "Short-term precipitation forecast based on the PERSIANN system and LSTM recurrent neural networks," *Journal of Geophysical Research: Atmospheres*, vol. 123, no. 22, nov 2018.

[118] X. Shi, Z. Chen, H. Wang, D.-Y. Yeung, W.-K. Wong, and W.-c. Woo, "Convolutional lstm network: A machine learning approach for precipi-

tation nowcasting," *Advances in neural information processing systems*, vol. 28, 2015.

[119] R. Sato, H. Kashima, and T. Yamamoto, "Short-term precipitation prediction with skip-connected PredNet," in *Artificial Neural Networks and Machine Learning – ICANN 2018*. Springer International Publishing, 2018, pp. 373–382.

[120] T. Song, W. Ding, J. Wu, H. Liu, H. Zhou, and J. Chu, "Flash flood forecasting based on long short-term memory networks," *Water*, vol. 12, no. 1, p. 109, dec 2019.

[121] M. G. Erechtchoukova, P. A. Khaiter, S. Saffarpour, S. chen, and M. Heralall, "Short-term prediction of flood events in a small urbanized watershed using multi-year hydrological records," in *The 21st International Congress on Modelling and Simulation (MODSIM2015)*, 2015.

[122] M. D. M.G. Erechtchoukova, P.A. Khaiter and D. Khaiter, "Role of a 'combination rule' in hybrid short-term prediction of hydrological events," in *22nd International Congress on Modelling and Simulation, Hobart, Tasmania, Australia*, 2017.

[123] M. G. Erechtchoukova and P. A. Khaiter, "The effect of data granularity on prediction of extreme hydrological events in highly urbanized watersheds: A supervised classification approach," *Environmental Modelling & Software*, vol. 96, pp. 232–238, oct 2017.

[124] S. Jackeray, A. S. Doradla, R. Rane, and B. Colaco, "A comparative review between programming tools used in data science," *IJCRT*, 2020.

[125] J. Brittain, M. Cendon, J. Nizzi, and J. Pleis, "Data scientist's analysis toolbox: Comparison of python, r, and sas performance," *SMU Data Science Review*, vol. 1, no. 2, p. 7, 2018.

[126] C. Ozgur, T. Colliau, G. Rogers, Z. Hughes, and E. Myer-Tyson, "Matlab vs. python vs. r," *Journal of data science*, vol. 15, pp. 355–371, 2021.

[127] Z. Wang, K. Liu, J. Li, Y. Zhu, and Y. Zhang, "Various frameworks and libraries of machine learning and deep learning: A survey," *Archives of Computational Methods in Engineering*, feb 2019.

[128] S. Bahrampour, N. Ramakrishnan, L. Schott, and M. Shah, "Comparative study of deep learning software frameworks," *arXiv preprint arXiv:1511.06435*, 2015.

[129] B. Lindemann, T. Müller, H. Vietz, N. Jazdi, and M. Weyrich, "A survey on long short-term memory networks for time series prediction," *Procedia CIRP*, vol. 99, pp. 650–655, 2021.

[130] R. Pascanu, T. Mikolov, and Y. Bengio, "On the difficulty of training recurrent neural networks," in *International conference on machine learning*. PMLR, 2013, pp. 1310–1318.

[131] R. Jozefowicz, W. Zaremba, and I. Sutskever, "An empirical exploration of recurrent network architectures," in *Proceedings of the 32nd International Conference on Machine Learning*, ser. Proceedings of Machine Learning Research, F. Bach and D. Blei, Eds., vol. 37. Lille, France: PMLR, 07–09 Jul 2015, pp. 2342–2350. [Online]. Available: https://proceedings.mlr.press/v37/jozefowicz15.html

[132] Z. C. Lipton, J. Berkowitz, and C. Elkan, "A critical review of recurrent neural networks for sequence learning," *arXiv preprint arXiv:1506.00019*, 2015.

[133] A. Botchkarev, "Evaluating performance of regression machine learning models using multiple error metrics in azure machine learning studio," *SSRN Electronic Journal*, 2018.

[134] A. Karpathy. (2019) a recipe for training neural networks_2019. [Online]. Available: http://karpathy.github.io/2019/04/25/recipe/#2-set-up-the-end-to-end-trainingevaluation-skeleton--get-dumb-baselines

[135] J. Sola and J. Sevilla, "Importance of input data normalization for the application of neural networks to complex industrial problems," *IEEE Transactions on Nuclear Science*, vol. 44, no. 3, pp. 1464–1468, 1997.

[136] S. S. Alahmari, D. B. Goldgof, P. R. Mouton, and L. O. Hall, "Challenges for the repeatability of deep learning models," *IEEE Access*, vol. 8, pp. 211 860–211 868, 2020.

Appendices

Appendix A

Experiments set

Table A.1: The flash flood prediction models performance before and after introducing future rainfall values.

Lead time	**Precision** **Mean** **Max** **Min** **SD**	**Recall** **Mean** **Max** **Min** **SD**	**F1** **Mean** **Max** **Min** **SD**	**Precision** **Mean** **Max** **Min** **SD**	**Recall** **Mean** **Max** **Min** **SD**	**F1** **Mean** **Max** **Min** **SD**
	Experiment set 1			Experiments set 2		
2013						
75	0.773	0.811	0.788	0.718	0.844	0.769
	0.870	0.880	0.851	0.889	0.960	0.841
	0.614	0.660	0.717	0.578	0.580	0.660
	0.071	0.048	0.035	0.079	0.084	0.045
90	0.711	0.769	0.733	0.769	0.831	0.795
	0.914	0.880	0.840	0.891	0.940	0.854

	0.545	0.620	0.661	0.594	0.700	0.689
	0.089	0.065	0.043	0.077	0.061	0.038
105	0.687	0.678	0.676	0.784	0.817	0.794
	0.893	0.800	0.753	0.925	0.940	0.854
	0.536	0.500	0.617	0.627	0.640	0.733
	0.092	0.059	0.039	0.075	0.076	0.033
120	0.508	0.489	0.493	0.790	0.805	0.789
	0.65	0.660	0.593	0.941	0.940	0.851
	0.414	0.320	0.368	0.672	0.320	0.478
	0.060	0.083	0.053	0.060	0.112	0.065
2014						
75	0.875	0.926	0.895	0.899	0.967	0.930
	0.947	1	0.947	1	1	0.974
	0.563	0.737	0.706	0.826	0.842	0.889
	0.091	0.053	0.052	0.046	0.049	0.024
90	0.836	0.807	0.818	0.863	0.968	0.910
	1	0.947	0.895	1	1	1
	0.692	0.636	0.686	0.731	0.684	0.788
	0.072	0.079	0.056	0.057	0.064	0.041
105	0.827	0.758	0.787	0.823	0.961	0.884
	1	0.895	0.865	0.947	1	0.950
	0.650	0.579	0.667	0.633	0.842	0.776
	0.082	0.075	0.054	0.074	0.046	0.043
120	0.823	0.660	0.726	0.826	0.949	0.881
	1	0.737	0.848	0.944	1	0.927
	0.500	0.421	0.558	0.613	0.842	0.760

	0.112	0.074	0.069	0.068	0.040	0.037
2015						
75	0.953	0.813	0.875	0.979	0.785	0.868
	1	0.935	0.935	1	0.935	0.951
	0.879	0.645	0.784	0.923	0.516	0.681
	0.037	0.076	0.036	0.027	0.082	0.051
90	0.963	0.772	0.852	0.974	0.772	0.858
	1	0.903	0.949	1	0.871	0.931
	0.794	0.355	0.512	0.813	0.645	0.764
	0.045	0.106	0.076	0.046	0.075	0.047
105	0.9356	0.751	0.831	0.942	0.813	0.869
	1	0.839	0.897	1	0.935	0.931
	0.839	0.613	0.760	0.778	0.677	0.792
	0.038	0.053	0.032	0.059	0.068	0.034
120	0.875	0.708	0.780	0.947	0.800	0.863
	0.955	0.806	0.862	1	0.935	0.915
	0.781	0.581	0.691 9	0.875	0.613	0.745
	0.049	0.068	0.041	0.043	0.087	0.047